2006-05-15

Food Shelf Life: Estimation and Experimental Design

Ross Allen Andrew Larsen
Brigham Young University - Provo

FOOD SHELF LIFE: ESTIMATION AND EXPERIMENTAL DESIGN

By

Ross A. A. Larsen

A project submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

August 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a project submitted by

Ross A. A. Larsen

This project has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

_____	_____
Date	G. Bruce Schaalje, Chair
_____	_____
Date	John S. Lawson
_____	_____
Date	Scott D. Grimshaw

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the project of Ross A. A. Larsen in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

_____ _____

Date G. Bruce Schaalje
Chair, Graduate Committee

Accepted for the Department

_____

Scott D. Grimshaw
Graduate Coordinator

Accepted for the College

_____

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

FOOD SHELF LIFE: ESTIMATION AND EXPERIMENTAL DESIGN

Ross A. A. Larsen

Department of Statistics

Masters of Science

Shelf life is a parameter of the lifetime distribution of a food product, usually the time until a specified proportion (1-50%) of the product has spoiled according to taste. The data used to estimate shelf life typically come from a planned experiment with sampled food items observed at specified times. The observation times are usually selected adaptively using 'staggered sampling.' Ad-hoc methods based on linear regression have been recommended to estimate shelf life. However, other methods based on maximizing a likelihood (MLE) have been proposed, studied, and used. Both methods assume the Weibull distribution. The observed lifetimes in shelf life studies are *censored,* a fact that the ad-hoc methods largely ignore. One purpose of this project is to compare the statistical properties of the ad-hoc estimators and the maximum likelihood estimator. The simulation study showed that the MLE methods have higher coverage than the regression methods, better asymptotic properties in regards to bias, and have lower median squared errors (mese) values, especially when shelf life is defined by smaller

percentiles. Thus, they should be used in practice. A genetic algorithm (Hamada et al. 2001) was used to find near-optimal sampling designs. This was successfully programmed for general shelf life estimation. The genetic algorithm generally produced designs that had much smaller median squared errors than the staggered design that is used commonly in practice. These designs were radically different than the standard designs. Thus, the genetic algorithm may be used to plan studies in the future that have good estimation properties.

ACKNOWLEDGEMENTS

I want to give my thanks to Dr. Bruce Schaalje for his many hours of advice, review, and direction. This project would not have been possible without his help. I would also like to acknowledge the support of my family. Their support has been invaluable. Finally, I would like to acknowledge the blessing of my Heavenly Father, who has opened many doors and given me much more than I could ever repay.

Contents

Tables

Table

Figures

Figures

Chapter 1

Introduction

Shelf life estimation is an important topic in food science (Labuza and Schmidl 1988). Shelf life is a parameter of the lifetime distribution of a food product, usually the time until a specified proportion (1-50%) of the product has spoiled according to taste (Freitas et al. 2003). The data used to estimate shelf life typically come from a planned experiment with sampled food items observed at specified times. The observation times are often selected adaptively using 'staggered sampling' (Labuza and Schmidl 1988, Gacula 1975). The observations are binary, indicating whether the sampled food items are acceptable or not at the observation times.

There are conflicting opinions regarding the statistical estimation of shelf life using such data. Ad-hoc methods based on linear regression have been recommended (Gacula and Singh 1984) and are still used (Cardelli and Labuza 2001). However, other methods based on maximizing a likelihood have been proposed, studied, and used (Freitas, Wagner, and Ho 2003, Gambaro, Fiszman, Gimenex, Varela, and Salvador

2004). To add to the confusion, some researchers use ad-hoc methods but incorrectly refer to them as maximum likelihood methods (Labuza and Schmidl 1988).

The observed lifetimes in shelf life studies are *censored* (Klein and Moeshcenberger 1997) because they are not the actual lifetimes. It is interesting to note that in medical studies, where the time of death of the patients is observed, observations can be either uncensored or censored. Censoring is usually rare, often due to dropout. However, in food shelf life studies all observations are censored. Unfortunately, the ad-hoc methods ignore the fact that the data are censored (Freitas et al. 2003). Additionally, some ad-hoc methods eliminate data beyond a certain point (Labuza and Cardelli 2000). In spite of these problems, shelf life estimates based on these methods often seem reasonable.

One purpose of this project is to compare the statistical properties of the ad-hoc estimators and the maximum likelihood estimator. This will be done through simulation of data from shelf life experiments. An additional purpose of the project is to develop a genetic algorithm (Hamada, Martz, Reese, and Wilson 2001) to find the near optimal staggered designs for estimating shelf life in specific situations. The genetic algorithm uses the simulation code to evaluate designs.

Chapter 2

Literature Review

Methods and study designs for estimating shelf life (Freitas et al. 2003) are based on lifetime models and survival analysis (Klein and Moeshcenberger 1997, Gacula and Singh 1984).

2.1 Lifetime Models

Let the time until spoilage or failure of a particular item of food be denoted as X. Since X varies from item to item let $f(x)$ be the probability density function of X and $F(x)$ be the corresponding cumulative distribution function of X. The survival function of X, denoted as $S(x)$, is the probability that $X>x$ and therefore $S(x) = 1–F(x)$. Another useful function when dealing with shelf life is the hazard function, defined as

$$h(x) = \frac{f(x)}{S(x)}. \qquad (1)$$

To understand the usefulness of the hazard function, note that $h(x)dx$ is the conditional probability $p(x < X < x + dx \mid X \geq x)$. The cumulative hazard function is defined as

$$H(x) = \int_0^x h(t)dt. \quad (2)$$

It turns out that

$$S(x) = e^{-H(x)}. \quad (3)$$

The shelf life of a food product is one of several parameters of the failure time distribution (Freitas et al. 2003). For example, $F^{-1}(.5)$ or $F^{-1}(.1)$, the fiftieth or tenth percentile, is often used as the shelf life (Cardelli and Labuza 2001, Freitas et al. 2003). The choice of percentile for shelf life is product- and research-specific.

2.2 Censoring and Design of Shelf Life Studies

Shelf life studies in food science are designed with planned observation times, $x_1, x_2, ..., x_n$, at which a panel of tasters judge whether each of a sample of food items is acceptable. This design generates only censored observations (Klein and Moeschenberger 1997), as the actual failure time of a food item is not directly observed. This testing is often destructive; that is, each food item can be sampled only once. For food item j and sampling time x_i, destructive sampling produces knowledge only of whether $X_j < x_i$ or $X_j > x_i$. If $X_j < x_i$ the observation is said to be left-censored; if $X_j > x_i$ the observation is right-censored. When sampling is not destructive and the food item can be resampled, the observations may also be interval-censored, that is $x_i < X_j < x_{i'}$ where $i \neq i'$.

‘Staggered sampling’ (Gacula 1975, Labuza and Schmidl 1988) is used to determine when and how many food items to sample. Staggered sampling is similar to the ‘Bruceton method’ (Wild and Collani, 2002) used in explosives research.

There are several variants of the staggered design, one of which was used by Labuza and Schmidl (1988). This staggered design used specific values of the parameters in Table 1.

Table 1: Parameters in staggered sampling

Parameter	Description	Values
n_0	Initial sample size	8
t_0	Time of first sampling	0
c	Constant increase in sample size from time *i to i+1*	1
? $_1$	Time between samples before first failure	3
? $_2$	Time between samples after first failure	3
? $_3$	Time between samples during acceleration phase	1
a	Proportion of failures used to define acceleration phase begins	.5

Let n_t be the number of food items tested at time t and f_t be the number that failed at time t. Both sample size and time are determined and incremented in the initial phase as follows:

$$n_{t_0} = n_0 \tag{4}$$

and

$$n_{t+\Delta_1} = n_t + c\,. \tag{5}$$

The parameter c is usually 0 or 1 because of limitations on the number of samples available. After the first food item has failed the time increment is shortened so that

$$n_{t+\Delta_2} = n_t + c\,. \tag{6}$$

Generally, $?_2 < ?_1$, although not in the design used by Labuza and Schmidl (1988). The acceleration phase begins once $100 \times a\%$ of the food items fail. During the acceleration phase, sample size and time are accelerated more quickly so that

$$n_{t+\Delta_3} = n_t + c + f_t . \tag{7}$$

2.3 The Weibull Distribution

The most common distribution used in food science for modeling lifetimes and estimating shelf life is the Weibull distribution (Gacula and Singh 1984). The Weibull distribution can be justified theoretically because of the concept of the weakest link in the molecular structure of food. The Weibull distribution has density function

$$f(x) = \left(\frac{b}{a^b} \right) x^{b-1} \exp\left[-\left(\frac{x}{a} \right)^b \right], \; x > 0 \tag{8}$$

and cumulative distribution function

$$F(x) = 1 - \exp\left[-\left(\frac{x}{a} \right)^b \right]. \tag{9}$$

The p^{th} percentile is

$$F^{-1}(p) = e^{\ln a} [-\ln(1-p)]^{\left(\frac{1}{b}\right)}. \tag{10}$$

Using (1) the hazard function of the Weibull distribution is

$$h(x) = \left(\frac{b}{a^b} \right) x^{b-1}. \tag{11}$$

It can be shown using (3) and (9) that the cumulative hazard is

$$H(x) = \left(\frac{x}{a} \right)^b . \tag{12}$$

Taking the natural log of both sides of (12) yields

$$\ln H(x) = b \ln x - b \ln a \,. \tag{13}$$

This is an important result because it is used both in assessing the fit of the Weibull distribution (Klein and Moschenberger 1997) and as the basis of a regression method to estimate the parameters of the distribution. The estimation method is called 'linear rectification' in mechanical engineering (Tobias and Trindade 1995). Food scientists usually rearrange (13) as

$$\ln X = \ln a + \left(\frac{1}{b}\right) \ln H(x) \tag{14}$$

and use linear regression based on (14) to find estimates of a and b .

2.4 Estimation of Shelf Life

Food scientists have used the modified linear rectification method (14) as a way to estimate the Weibull parameters (Gacula and Singh 1984, Labuza and Cardelli 2000). The right-censored observations are ignored. The times for the left-censored observations are ordered, and the reverse ranks of those values are computed. Decreasing ranks are assigned to tied observations as if they were ordered meaningfully rather than arbitrarily.

Next the hazard *h(x)* from (11) for each left- or interval-censored data point is estimated as the reciprocal of the corresponding reverse rank. The cumulative hazard *H(x)* from (12) is then calculated by summing the *h(x)* estimates across preceding data points. This is similar to the Nelson-Aalen estimator of *H(x)* (Klein and Moeschberger 1997) except that the right-censored and tied data are ignored and the left- and interval-censored data are treated as uncensored observations. Logarithms are then calculated for *X* and *H(x)*, and simple linear regression based on (14) is used to obtain estimates of *ln(a)*

and *1/ß* (the intercept and slope, respectively). This initial estimation technique will be called the 'Gacula' method throughout this paper.

Some difficulties inherent in the estimation of the Weibull parameters using (14) are acknowledged by food scientists and several ad-hoc correction measures have been suggested. One ad-hoc correction method is to ignore the data that have a cumulative hazard greater than 100 (Cardelli and Labuza 2000). This method will be called the 'Labuza' method in this paper. The method is rationalized as a way to force the shape parameter of the Weibull distribution in a certain range, specifically, $2 < b < 4$.

Sloan (2005) proposed recalculating the cumulative hazard once the data with a cumulative hazard greater than 100 have been removed. This method will be called the 'Sloan' method.

Once *ln(a)* and *1/ß* have been estimated by linear regression based on (14), the appropriate percentile (shelf life) is estimated by inserting *ln(a)* and *1/ß* into (10).

2.5 Maximum Likelihood Estimation of Shelf Life

Maximum likelihood techniques for censored lifetime data are well established (Klein and Moeschberger 1997). Freitas et al. (2003) and Gambaro et al. (2004) suggested the use of these techniques for shelf life estimation. The likelihood for left and right-censored data from the Weibull distribution is (Freitas et al. 2003)

$$L(q) = \prod_{i=1}^{k} \prod_{j=1}^{n_i} \left[e^{-(x_i a)^b} \right]^{1-y_{ij}} \left[1 - e^{-(x_i a_j)^b} \right]^{y_{ij}}, \qquad (15)$$

where *i* denotes the sampling time, *j* denotes the food item, and

$$y_{ij} = \begin{cases} 1 \text{ if } 0 < X_{ij} < x_i \\ 0 \text{ if } X_{ij} > x_i \end{cases}.$$

Thus, the likelihood method handles left- and right-censored data as defined in section 2.2.

This method can be altered to allow for interval censoring when sampling is nondestructive. Let x_{li} be the last time item i was acceptable and x_{ui} be the time when item i was observed to have spoiled. The likelihood is then:

$$L(a,b)=\prod_{i=1}^{n}\left[e^{-(x_{li}a)^b}\right]^L\left[1-e^{-(x_{ui}a)^b}\right]^R\left[e^{-(x_{li}a)^b}-e^{-(x_{ui}a)^b}\right]^I \tag{16}$$

where

$$L=\begin{cases}1 \text{ if } x_{li}=0 \text{ and } 0<x_{ui}<\infty\\ 0 \text{ otherwise}\end{cases}$$
$$R=\begin{cases}1 \text{ if } 0<x_{li}<\infty \text{ and } x_{ui}=\infty\\ 0 \text{ otherwise}\end{cases}.$$
$$I=\begin{cases}1 \text{ if } 0<x_{li}<x_{ui}<\infty\\ 0 \text{ otherwise}\end{cases}$$

Thus, if the food item is sampled and not spoiled before the food item is depleted then (16) with R=1 in the exponent is used. If the food item is sampled only once and is spoiled then (16) with the L=1 exponent is used. Finally, if a food item is sampled repeatedly and fails before it is depleted then (16) with I=1 exponent is used.

An additional advantage of the maximum likelihood approach is that the effects of covariates on survival can be modeled using

$$a=\exp\{Z_i',q\}=\exp\{Z_j^0q_0+Z_j^1q_1+...+Z_j^qq_q\} \tag{17}$$

where $Z_j=(Z_j^0,Z_j^1,...,Z_j^q)$ is a vector of covariates for food item j and $q=(q_0,q_1,...,q_q)'$ is a vector of parameters (Freitas et al. 2003). This is sometimes referred to as the accelerated failure time model (Klein and Moeschberger 1997).

Maximum likelihood estimates are obtained by direct maximization of the logarithm of (15) or (16) (Freitas et al. 2003). Numerical optimization methods such as the Newton-Raphson algorithm are used to calculate the values. The standard errors for the estimates are found using large sample theory and the Delta method (Freitas et al. 2003). In order to estimate the shelf life percentiles, maximum likelihood estimates of the Weibull parameters are inserted into (10). Estimates thus obtained are also maximum likelihood estimates because of the property of functional invariance (Rencher 2000, p. 232). Asymptotic confidence intervals, based on Delta-method standard errors, have good coverage for percentiles close to 50%. However, asymptotic confidence intervals for the smaller percentiles are questionable as the normality assumption is violated (Pawitan 2001, p. 41-42). The percentile estimates and their standard errors are easily found in SAS using PROC LIFEREG, MINITAB, or SPLIDA in SPLUS.

One assumption for maximum likelihood techniques is independence of observations. In staggered sampling this is not the case as the design adapts depending on current observations. For the purposes of this project the maximum likelihood model that assumes independence will be used.

Chapter 3

Examples

The estimators discussed in Chapter 2 are illustrated by applying them to two real data sets.

3.1 Taro Root

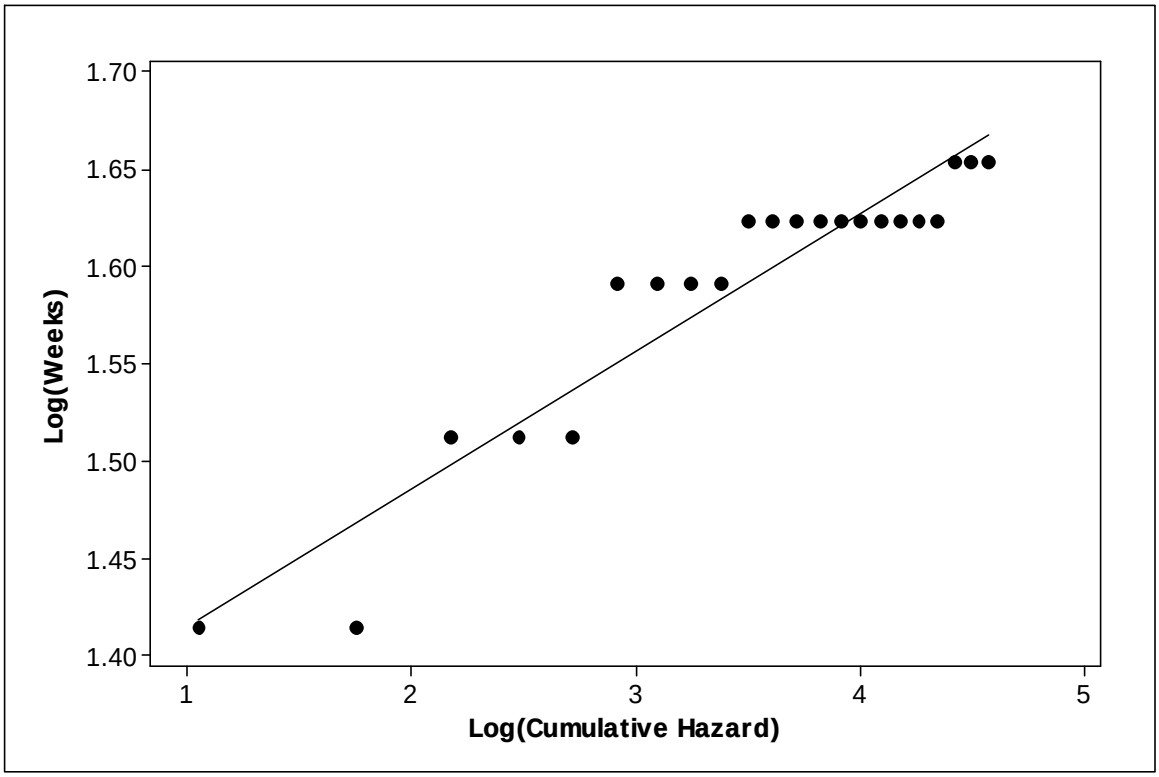
Sloan (2005) studied the shelf life of packaged taro root from Fiji. In one of his studies, the product was stored at 45^{o} Celsius (Table 2). A staggered design was used where n_0 was 4, c was 1, τ_1 was fourteen days and τ_2 was six days. The a parameter was chosen to be 50% and τ_3 was chosen to be three (see section 2.2). The design was not followed strictly due to weekends.

Table 2: Shelf life data for taro root kept at 45 degrees

Weeks of Storage	Number tested	Number failed	Percent Failed
15.0	4	0	.00
26.0	5	2	.40
32.5	6	3	.50
39.0	10	4	.40
42.0	15	10	.67
45.0	23	15	.65

The plot of the log of the cumulative hazard versus the log of weeks (section 2.4) is relatively linear (Figure 1) which suggests that the Weibull distribution fits the data well.

Figure 1: Log of weeks versus log of the cumulative hazard for taro root



Shelf life estimates and standard errors (Table 3) were obtained using SAS PROC REG and SAS PROC LIFEREG (SAS Institute Inc 1999) using the code in Appendix 1. Estimates based on linear regression (Gacula and Singh 1984, Labuza and Schmidl 1985,

Sloan 2005) agreed fairly well with each other but were all larger than the maximum likelihood estimates. This difference is accentuated for the lower percentile estimates. Also, the standard errors were much smaller for the linear rectification methods. The Sloan estimates were closest to the maximum likelihood estimates.

Table 3: Percentile estimates for taro shelf life

Method	1^{st}(SE)	10^{th}(SE)	25^{th}(SE)	Median(SE)
Gacula	25.76(1.07)	30.97(0.68)	33.60(0.50)	41.58(0.47)
Labuza	22.16(0.96)	32.42(0.56)	38.15(0.43)	43.99(0.65)
Sloan	22.05(1.43)	29.95(0.95)	34.13(0.69)	38.28(0.63)
MLE	4.71(5.05)	14.86(7.54)	24.28(6.57)	37.33(3.58)

3.2 Coffee

Cardelli and Labuza (2000) studied the shelf life of coffee (Table 4). A staggered design was used where n_0 was 4, c was 1, $?_1$ was seven weeks and $?_2$ was five weeks. The a parameter was chosen to be 50% and $?_3$ was chosen to be three weeks (see section 2.2). The design was not followed strictly.

Table 4: Shelf life data of coffee

Weeks of Storage	Number tested	Number failed	Percent Failed
7.1	4	1	.25
12.1	5	1	.20
17.3	6	4	.67
20.1	11	7	.64
23.3	19	12	.63

Because a plot of the log of cumulative hazard versus the log of weeks is linear (Figure 2) the Weibull distribution seems reasonable (Gacula and Singh 1984).

Figure 2: Log of weeks and log of cumulative hazard for coffee

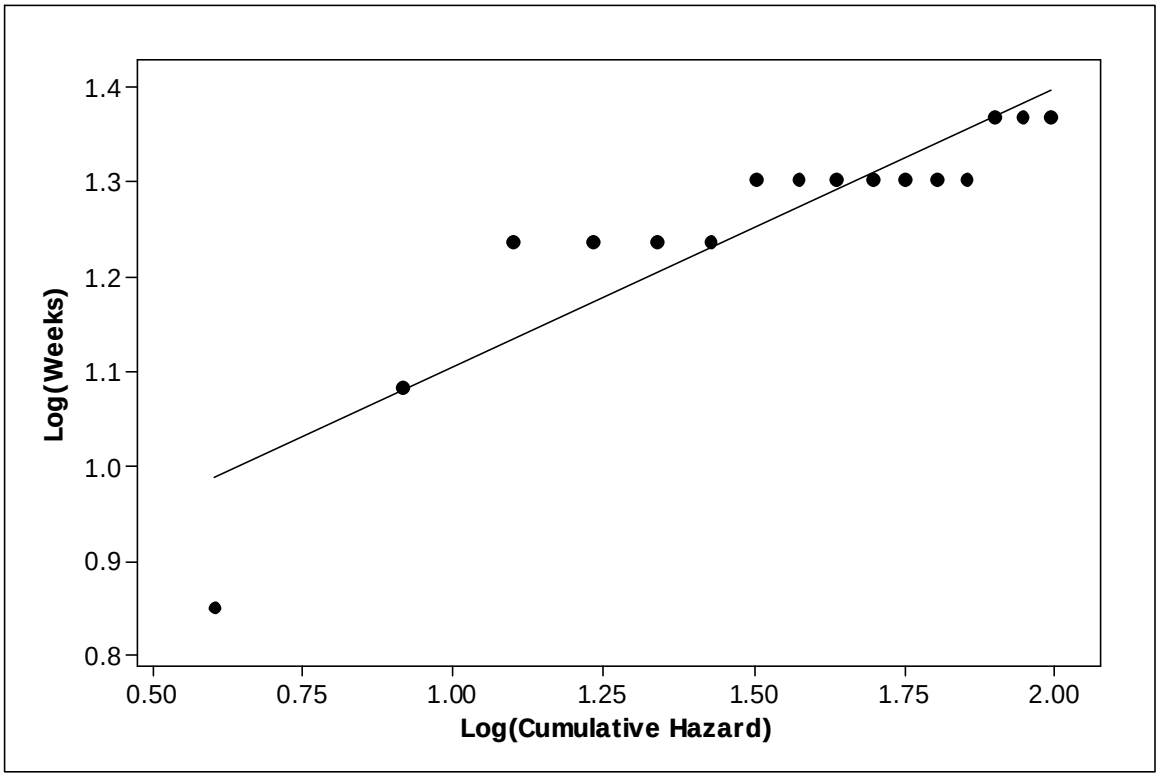


Table 5: Percentile Estimates for coffee shelf life

Method	1^{st}(SE)	10^{th}(SE)	25^{th}(SE)	Median(SE)
Gacula	8.69(0.92)	12.05(0.72)	17.02(0.57)	20.30(0.57)
Labuza	6.49(0.85)	12.93(0.68)	17.37(0.57)	22.48(0.95)
Sloan	6.76(1.01)	11.81(0.86)	14.99(0.69)	18.47(0.68)
MLE	0.65(1.30)	3.90(3.75)	8.40(4.39)	16.44(3.12)

The same trends that are present in the Taro data are also present in the coffee example. Also, it is interesting to note that the data suggests that the 50^{th} percentile occurs on or before 17 weeks of storage (Table 4) and yet the linear rectification techniques estimate the percentile as being much higher.

Chapter 4

Methods

4.1 Simulation Study

In order to compare the regression estimators of shelf life (Gacula and Singh 1984, Labuza and Schmidl 1988), and the maximum likelihood estimator, a simulation study was performed. This was done in SAS using procedures IML, REG, and LIFEREG (SAS Institute Inc. 1999). The SAS code can be found in Appendix 2. Each run of the simulation was done on an Optiplex GX260 and GX270 Dwll computer, running windows XP with SAS 9.0.

Several different scenarios were considered. Simulation data were drawn from various Weibull distributions (Figure 3) with specified parameter values (Table 6) using staggered sampling (Table 7), but only destructive sampling was considered. Each of the scenarios were simulated 10,000 times. For each simulation, various percentiles were estimated using the different techniques. Distributions of the estimates were examined and compared.

Figure 3: Four Weibull distributions used in the simulations

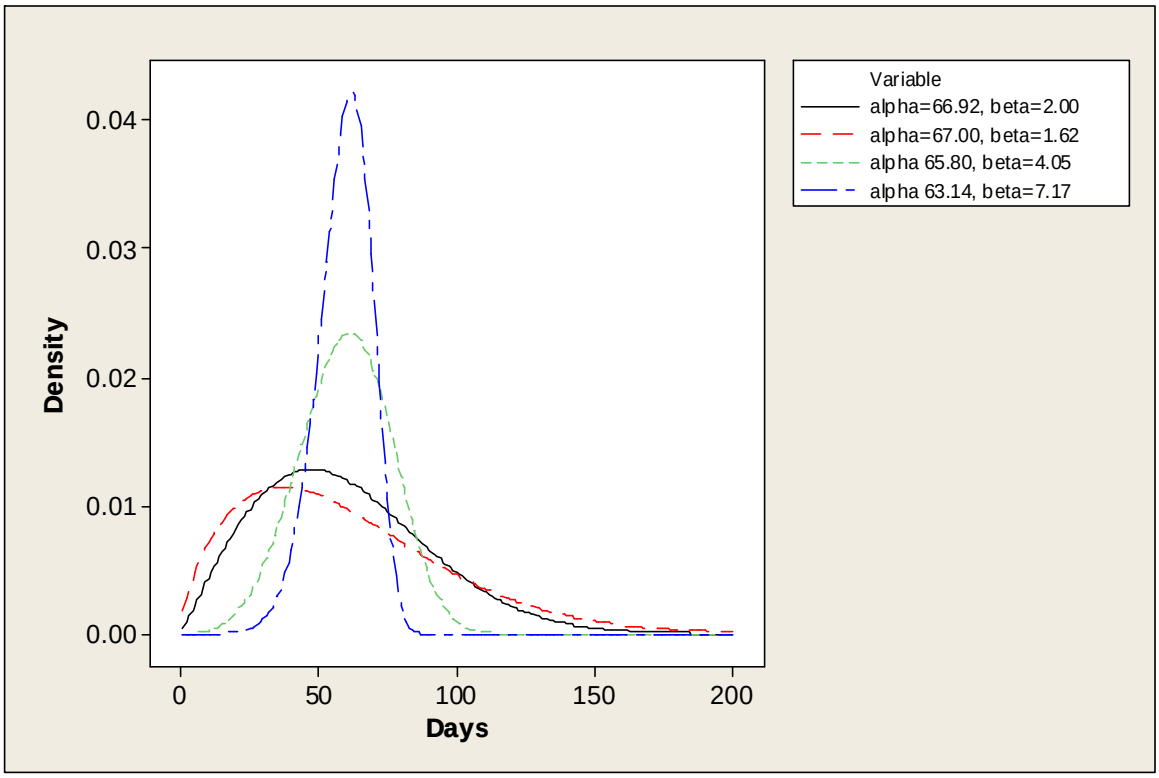


Table 6: Variables of the simulation studt

Parameter	Description
a	Centrality parameter for the Weibull distribution
b	Shape parameter for the Weibull distribution
N	Total sample size

Table 7: Parameters for the staggered sampling design used in the simulation study

Parameter	Description	Value
n_0	Initial sample size	3
t_0	Time of first sampling	7
c	Constant increase in sample size from time *i to i+1*	1
? $_1$	Time between samples before first failure	14
? $_2$	Time between samples after first failure	7
? $_3$	Time between samples during acceleration phase	3
a	Proportion of failures used to define acceleration phase begins	.5

Thus, the simulation study was a 4x2x2 factorial with 4 Weibull distributions, 2 sample sizes, and 2 sampling plans (independent or staggered). The four Weibull

distributions had a constant mean of 60, but the shape and the variance of the distributions varied (Figure 3). The mean of 60 was arbitrary but does not affect the generality of the results because data may be scaled in any way the researcher wishes. These distributions were similar to those found in food science literature (Gacula and Singh 1984). Two distributions were approximately symmetrical or 'less skewed' but one has a large variance and one has a small variance. Two distributions were 'more skewed', one with a large variance and one with a small variance. The total sample size (N) varied from 50 to 75, both of which are typical values in food science literature. Additionally, one non-adaptive sampling plan with values contained in Table 8 was used to show how the techniques perform when the independence assumption is not violated.

Table 8: Parameters for the non-adaptive (independent) sampling designs used in the simulation study

Parameter	Description	Value
n_0	Initial sample size	7
t_0	Time of first sampling	7
c	Constant increase in sample size from time *i to i+1*	0
? $_1$	Time between samples before first failure	7
? $_2$	Time between samples after first failure	7
? $_3$	Time between samples during acceleration phase	7
a	Proportion of failures used to define acceleration phase begins	NA

The estimates for 1st, 10th, 25th, and 50th percentiles were computed using the same four methods as in the examples. Distributions of estimates relative to true shelf life values were summarized and displayed using bias, median squared error (mese) and coverage percentages of varying percentiles of confidence intervals. In certain randomly generated data sets 2 or fewer food items were left-censored. This causes complications in the estimation methods. A researcher would normally not use such data to form an

estimate. Because of the unrealistic nature of such studies, such cases were excluded in the analysis.

4.2 Near Optimal Design using a Genetic Algorithm

The food science literature has minimal guidance on efficient design for shelf life estimation studies. The generally accepted staggered sampling technique (Gacula 1975, Cardelli and Labuza 2001) may be optimal to estimate the median. Still it lacks theoretical backing and instructions on how to alter the scheme to estimate percentiles other than the median. To address this problem, the simulation developed in the first part of this project was used in connection with a genetic algorithm (Hamada et al. 2001) in finding near-optimum sampling designs for general shelf life estimation using the MLE.

Under the genetic algorithm, the simulation was run many times with varying staggered sampling parameter values (Table 8). On each simulation run, an optimality measure was computed and used to compare different sampling schemes. Because of the theoretical superiority of the MLE, it was used to calculate the estimate. The optimality measure was the mese of the maximum likelihood estimates. The genetic algorithm searched the design space and selected a near optimal solution (Hamada et al. 2001). This solution contained specific parameter values from specified sets of possible values (Table 9).

Table 9: Parameters and design space

Parameter	Description	Possible Values
n_0	Initial sample size	1,2,…,25
t_0	Time of first sampling	1,2,…,25
c	Constant increase in sample size from time *i to i+1*	0,1,…,5
? $_1$	Time between samples before first failure	$F^{-1}(.01), F^{-1}(.01)+1,\ldots,F^{-1}(.5)-1,F^{-1}(.5)$*
? $_2$	Time between samples after first failure	1,2,…, $F^{1}(.5)- F^{-1}(.01)$*
? $_3$	Time between samples during acceleration phase	1,2,…, $F^{-1}(.5)- F^{-1}(.1)$*
a	Percentage of failure when acceleration phase begins	1,2,…,50%

*percentiles of specified Weibull distribution, rounded to whole numbers

The genetic algorithm was developed using the SAS MACRO language (SAS Institute 1997). The user specifies parameters of the Weibull distribution, the shelf life percentile of interest, the total sample size (N), the number of sampling designs generated at each step of the genetic algorithm (M), the number of generations for the genetic algorithm (G), and the number of iterations to be performed for each simulation per sampling design (S). Ideally, S should be large, such as 10,000; nevertheless computing resource limitations and computational errors in PROC LIFEREG made it expedient to keep S smaller.

Pseudo-code of the genetic algorithm used to find the near-optimal sampling scheme (Hamada et al. 2001) is as follows.

GENERATION 0:

- Generate *M* random sampling schemes by randomly selecting values for the design parameters (Table 9)
- Use the simulation code to generate S iterations for each sampling design
- Calculate optimality for each of the sampling designs (mese)
- Order the designs by optimality

GENERATIONS g=1,…,G:

- Generate *M* sampling schemes by CROSSOVER (see below)

- Generate *M* sampling schemes by MUTATION (see below)
- Evaluate optimality of the 2*M* designs generated by crossover and mutation.
- Order 3*M* designs (2*M* designs by crossover and mutation and top *M* designs from generation *g*-1) by decreasing optimality
- Retain *M* designs with best optimality for generation *g*+1

CROSSOVER:

- From the *M* sampling designs retained from the previous generation, pick two sampling schemes with probability inversely proportional to their optimality rank (best optimality has rank 1)
- The *i*th parameter (Table 8) of the generated sampling design is realized by randomly choosing from the *i*th variable of the two picked sampling schemes, i=1,..,*7*

MUTATION:

- For each of the *M* designs retained from the previous generation (referred to as a current schemes), a new scheme is generated as follows:

  Each parameter realization is mutated with probability *exp(-ug); u* is a tuning constant and the mutation probability decreases with increasing *g* (number of generations). The tuning constant *u* for the mutation algorithm was arbitrarily chosen to be 0.10.
- If an entry is mutated, the new entry is obtained by drawing from a particular distribution which is approximately centered at the current entry and whose variance depends on a tuning parameter ?. The chosen distributions were binomial for the discrete parameters, with the mean centered at the current parameter value. For the continuous parameter *a* the Beta distribution was used with arguments 2 and $\frac{2-2a}{a}$.

The genetic algorithm was tested using two Weibull distributions. One was less skewed with small variance (a =63, b =7) and one was more right skewed with large variance (a =67, b =2). The two typical sample sizes of 50 and 75 were considered, and destructive testing was used. Values of 50 and 5 were used for both G, and M, and S was set at 100 and 1000. Thus, the genetic algorithm was tested at 2^5 =32 total settings. Effects of G, M, and S on convergence of the algorithm were demonstrated using box plots of mese values. Also, a measure of overall performance of the algorithm was defined as the 'improvement ratio'

$$1-\frac{\text{Best mese}_{\text{Generation G}}}{\text{Average mese}_{\text{Generation 0}}}. \qquad (18)$$

Chapter 5

Results

This chapter presents results of the simulation study and reports on development of the genetic algorithm. Coverage of .50, .25, .10, and .01 percentiles confidence intervals, mese, and bias are presented comparing the regression methods to the maximum likelihood method. Mese and the ratio of improvement are used to evaluate the effectiveness of the genetic algorithm.

5.1 Coverage of 95% confidence intervals

The MLE techniques produced 95% confidence intervals with high coverage but not consistently near the target of 95% coverage (Table 10). For staggered designs this may be partly due to the non-independence of the data. However, coverage for even the independent designs was not always near 95%, especially for the more skewed distributions.

Table 10: Coverage of 95% confidence intervals for Gacula, Labuza, Sloan, and MLE methods with N=50

Percentile	Design	Shape	Variance	coverage of 95% intervals			
				Gacula	Labuza	Sloan	MLE
0.50	independent	less skewed	small	0.13	0.28	0.02	0.86
			large	0.21	0.33	0.05	0.80
		more skewed	small	0.17	0.30	0.06	0.82
			large	0.17	0.28	0.06	0.82
	staggered	less skewed	small	0.10	0.21	0.03	0.84
			large	0.03	0.06	0.00	0.81
		more skewed	small	0.02	0.02	0.02	0.76
			large	0.09	0.10	0.06	0.80
0.25	independent	less skewed	small	0.13	0.11	0.00	0.86
			large	0.52	0.58	0.21	0.93
		more skewed	small	0.43	0.35	0.51	0.94
			large	0.56	0.58	0.53	0.94
	staggered	less skewed	small	0.02	0.02	0.05	0.88
			large	0.06	0.08	0.01	0.82
		more skewed	small	0.27	0.25	0.23	0.79
			large	0.07	0.09	0.02	0.80
0.10	independent	less skewed	small	0.51	0.50	0.41	0.98
			large	0.51	0.54	0.62	0.99
		more skewed	small	0.07	0.05	0.17	0.89
			large	0.13	0.12	0.25	0.90
	staggered	less skewed	small	0.01	0.01	0.01	0.88
			large	0.01	0.01	0.05	0.84
		more skewed	small	0.07	0.07	0.06	0.79
			large	0.01	0.01	0.00	0.80
0.01	independent	less skewed	small	0.78	0.80	0.88	0.99
			large	0.35	0.41	0.39	0.89
		more skewed	small	0.05	0.07	0.06	0.83
			large	0.10	0.11	0.10	0.86
	staggered	less skewed	small	0.05	0.05	0.06	0.88
			large	0.02	0.02	0.02	0.85
		more skewed	small	0.03	0.04	0.04	0.78
			large	0.03	0.03	0.03	0.79

The best coverage for the MLE method occurred when estimating the 10^{th} percentile for the independent designs. This may be due to the nature of this particular independent design (Table 8) where more data is gathered near the true 10^{th} percentile. The worst coverage occurred at the 50^{th} percentile. In spite of the fact that MLE coverage is not near the target of 95%, it greatly outperformed the regression methods. The poorest MLE coverage was 76% while the regression techniques consistently had very low coverage.

The regression techniques had very small standard errors for the example data sets (Tables 3 and 5), and thus their corresponding confidence intervals were narrow. This was reflected in their poor coverage of the true shelf life; hence the small standard errors for the regression methods are illusionary and do not reflect the true uncertainty of the estimates (Tables 10). However, there were times when coverage of the regression techniques did approach 95% such as when the estimating the 1^{st} percentile with the less-skewed small-variance distribution using the independent design.

Table 11: Coverage of 95% confidence intervals for Gacula, Labuza, Sloan, and MLE methods with N=75

Percentile	Design	Shape	Variance	coverage of 95% intervals			
				Gacula	Labuza	Sloan	MLE
0.50	independent	less skewed	small	0.14	0.02	0.50	0.94
			large	0.63	0.37	0.68	0.94
		more skewed	small	0.51	0.71	0.36	0.92
			large	0.46	0.65	0.30	0.91
	staggered	less skewed	small	0.32	0.14	0.36	0.89
			large	0.28	0.25	0.16	0.88
		more skewed	small	0.01	0.02	0.01	0.85
			large	0.01	0.01	0.00	0.82
0.25	independent	less skewed	small	0.05	0.02	0.14	0.92
			large	0.11	0.07	0.22	0.93
		more skewed	small	0.07	0.02	0.16	0.93
			large	0.02	0.01	0.09	0.93
	staggered	less skewed	small	0.04	0.05	0.02	0.89
			large	0.20	0.18	0.17	0.87
		more skewed	small	0.20	0.20	0.16	0.83
			large	0.05	0.04	0.08	0.82
0.10	independent	less skewed	small	0.04	0.04	0.11	0.91
			large	0.04	0.04	0.09	0.92
		more skewed	small	0.00	0.00	0.01	0.92
			large	0.00	0.00	0.00	0.92
	staggered	less skewed	small	0.07	0.06	0.06	0.89
			large	0.02	0.02	0.01	0.88
		more skewed	small	0.03	0.03	0.03	0.84
			large	0.02	0.01	0.05	0.82
0.01	independent	less skewed	small	0.07	0.14	0.13	0.92
			large	0.05	0.09	0.08	0.92
		more skewed	small	0.00	0.01	0.01	0.89
			large	0.00	0.00	0.00	0.88
	staggered	less skewed	small	0.03	0.04	0.04	0.90
			large	0.04	0.04	0.04	0.88
		more skewed	small	0.02	0.03	0.02	0.82
			large	0.00	0.00	0.00	0.80

Simulations with the larger sample size of 75 confirmed the assumed asymptotic normality basis of the MLE confidence intervals. In other words, as the sample size increased the confidence interval coverage approached 95% (Tables 10 and 11). Asymptotic normality is also consistent with the result that for independent designs, coverages for interval estimates of the 50^{th} and 25^{th} percentiles were closer to the target than for interval estimates of the 10^{th} and 1^{st} percentiles. This is due to the sampling distributions being bounded by zero. Overall, the coverage for the sample size of 75 was greatly improved for the MLE techniques.

Additionally, the MLE-based confidence intervals again greatly outperformed those of the regression techniques. The coverage for regression techniques, while being relatively high for the 50^{th} percentile, was lower overall with the higher sample size of 75. This was especially true when estimating the 1^{st} percentile for the less-skewed small-variance distribution using the independent design. The contrast of these results to those for sample size 50 suggests that the occurrences of near target coverage for the regression methods are unpredictable.

5.2 Mese

Surprisingly, the mese values for the MLE technique were larger for the independent designs than the staggered designs for the 50^{th} percentile (Table 12). This was not consistent, however, over the various percentiles. For example, both designs were more or less equivalent for the 25^{th} percentile. The mese actually became better for the independent design as compared to the staggered design for the 10^{th} percentile and became much better for the 1^{st} percentile.

As compared to the regression techniques, mese values for the MLE techniques were not consistently smaller, especially for the 25th percentile. As the percentiles became smaller, the MLE mese values became smaller as compared to the mese values produced by the regression techniques. This was true especially for the 1st percentile where the differences were quite dramatic. Mese values for the regression techniques were generally smaller for the independent designs as opposed to the staggered designs.

Table 12: Mese values for Gacula, Labuza, Sloan, and MLE methods for estimating shelf life with N=50

Percentile	Design	Shape	Variance	mese			
				Gacula	Labuza	Sloan	MLE
0.50	independent	less skewed	small	69.7	95.7	144.2	33.7
			large	84.2	81.9	168.3	58.2
		more skewed	small	126.9	91.0	226.6	86.7
			large	127.0	81.9	201.5	96.8
	staggered	less skewed	small	9.2	6.0	14.8	6.2
			large	17.3	16.6	27.8	14.7
		more skewed	small	123.1	118.9	154.2	48.1
			large	119.6	110.3	259.5	65.1
0.25	independent	less skewed	small	28.6	24.3	62.0	16.9
			large	7.9	6.5	32.8	9.4
		more skewed	small	6.8	6.3	8.3	17.2
			large	12.3	19.0	7.2	17.0
	staggered	less skewed	small	9.9	11.7	6.4	6.9
			large	36.2	38.1	32.1	12.7
		more skewed	small	37.8	38.2	35.0	21.0
			large	88.4	87.9	84.2	27.3
0.10	independent	less skewed	small	8.2	8.2	13.2	14.6
			large	8.7	8.6	11.1	10.1
		more skewed	small	65.4	64.7	47.1	20.9
			large	101.3	100.2	78.3	23.5
	staggered	less skewed	small	53.8	52.9	48.1	15.1
			large	195.5	194.0	182.5	25.9
		more skewed	small	294.6	293.7	280.4	35.8
			large	233.4	233.4	169.5	54.2
0.01	independent	less skewed	small	18.0	28.1	23.8	22.8
			large	62.7	55.7	80.1	37.3
		more skewed	small	135.3	114.2	143.6	16.8
			large	125.0	100.4	125.2	9.7
	staggered	less skewed	small	224.9	198.4	218.0	50.1
			large	623.9	602.0	627.0	75.1
		more skewed	small	672.5	648.7	681.4	39.1
			large	407.8	337.8	367.8	15.3

Table 13: Mese values for Gacula, Labuza, Sloan, and MLE methods for estimating shelf life with N=75

				mese			
Percentile	Design	Shape	Variance	Gacula	Labuza	Sloan	MLE
0.50	independent	less skewed	small	23.1	32.5	5.1	2.4
			large	3.3	13.7	2.7	7.8
		more skewed	small	8.8	6.7	21.7	25.5
			large	13.6	6.9	35.0	34.6
	staggered	less skewed	small	2.8	4.2	2.0	3.0
			large	2.6	3.2	6.6	8.2
		more skewed	small	68.2	62.2	89.1	31.0
			large	107.5	89.3	168.9	39.4
0.25	independent	less skewed	small	46.5	47.8	24.5	3.7
			large	40.7	58.2	23.5	6.8
		more skewed	small	74.5	98.5	47.1	11.9
			large	93.7	120.9	62.6	13.2
	staggered	less skewed	small	29.6	33.0	22.9	4.7
			large	67.6	70.3	57.9	8.4
		more skewed	small	71.1	73.0	60.1	18.0
			large	52.5	51.0	71.8	20.5
0.10	independent	less skewed	small	75.6	71.3	56.3	8.3
			large	110.8	105.3	83.9	12.2
		more skewed	small	208.4	206.0	175.3	18.1
			large	242.7	239.6	207.6	17.6
	staggered	less skewed	small	91.2	89.8	81.4	11.3
			large	251.1	249.1	233.5	20.2
		more skewed	small	364.6	363.5	347.9	39.0
			large	255.0	257.0	239.3	44.6
0.01	independent	less skewed	small	130.6	108.6	130.5	17.6
			large	222.1	163.2	195.5	21.5
		more skewed	small	248.3	200.6	233.6	10.6
			large	224.7	183.3	211.4	5.7
	staggered	less skewed	small	273.4	243.3	261.2	30.1
			large	678.2	649.4	671.7	46.0
		more skewed	small	730.0	699.4	728.8	29.4
			large	439.8	402.9	435.3	14.7

With a larger N (Table 13) the superiority of the MLE method became apparent as the MLE mese values were consistently much smaller than the mese values for the regression techniques for all percentiles except the 50^{th} percentile. As expected, independent sampling displayed mese values which were smaller than those for the staggered design at all percentiles.

The regression techniques varied in whether the staggered or independent designs displayed smaller mese values. For example, in the 10^{th} and 1^{st} percentiles, the mese values were smaller for the independent designs than staggered designs for all the regression techniques. However, for the 50^{th} percentile, the Gacula and Labuza techniques did not consistently have smaller mese values for the independent designs.

5.3 Bias

The bias values for the MLE method were generally positive except for the less-skewed small-variance distribution using the independent design (Table 14). Bias for the regression techniques was generally negative for the 50^{th} percentile estimates, but positive for the smaller percentiles. Surprisingly, the MLE method did not consistently have lower bias than the regression methods. The smallest percentiles frequently displayed smaller bias for the MLE methods than the regression techniques, but for the 50^{th} and 25^{th} percentiles, bias values were frequently smaller for the regression technique.

Table 14: Bias for Gacula, Labuza, Sloan, and MLE methods for estimating shelf life with N=50

				bias			
Percentile	Design	Shape	Variance	Gacula	Labuza	Sloan	MLE
0.50	independent	less skewed	small	-8.3	-8.2	-12.4	-2.1
			large	-8.3	-7.0	-13.4	3.5
		more skewed	small	-10.3	-8.2	-14.8	8.3
			large	-10.4	-8.0	-14.3	9.0
	Staggered	less skewed	small	0.9	1.4	-0.3	4.0
			large	-4.9	-4.3	-6.5	7.0
		more skewed	small	-11.3	-10.7	-13.3	4.1
			large	-12.7	-12.1	-14.7	5.0
0.25	independent	less skewed	small	-5.4	-5.2	-7.7	-3.5
			large	-2.8	-2.1	-5.8	1.4
		more skewed	small	1.3	2.2	-1.3	4.2
			large	3.3	4.3	1.0	4.0
	Staggered	less skewed	small	5.6	5.8	4.7	3.2
			large	4.1	4.3	2.9	5.1
		more skewed	small	5.0	5.3	3.7	5.8
			large	5.9	6.3	4.7	5.2
0.10	independent	less skewed	small	-2.5	-2.0	-3.1	-4.4
			large	1.9	2.1	0.7	-0.3
		more skewed	small	8.2	8.2	6.9	1.3
			large	10.2	10.2	9.0	1.0
	Staggered	less skewed	small	10.2	10.1	9.5	2.1
			large	11.7	11.7	11.0	3.3
		more skewed	small	15.7	15.7	15.0	4.0
			large	16.7	16.7	16.0	4.0
0.01	independent	less skewed	small	2.5	3.9	5.2	-4.8
			large	8.1	8.0	9.5	0.3
		more skewed	small	11.8	10.9	12.1	1.9
			large	11.7	10.7	11.8	1.8
	Staggered	less skewed	small	17.9	17.3	17.8	1.7
			large	22.1	21.6	22.3	3.5
		more skewed	small	23.9	23.5	24.2	5.2
			large	22.7	22.3	22.9	4.9

Table 15: Bias for Gacula, Labuza, Sloan, and MLE methods for estimating shelf life with N=75

				bias			
Percentile	Design	Shape	Variance	Gacula	Labuza	Sloan	MLE
0.50	independent	less skewed	small	4.6	6.0	2.5	0.5
			large	1.4	3.7	-0.7	1.8
		more skewed	small	-2.2	0.8	-4.7	4.9
			large	-3.1	0.1	-5.6	5.9
	Staggered	less skewed	small	3.4	3.8	2.2	2.6
			large	-1.9	-1.2	-3.3	5.1
		more skewed	small	-8.5	-7.8	-10.0	6.9
			large	-9.8	-9.1	-11.2	7.0
0.25	independent	less skewed	small	6.7	7.3	5.0	0.5
			large	6.5	7.5	4.8	1.0
		more skewed	small	8.7	9.9	6.9	1.4
			large	9.9	11.2	8.2	1.3
	Staggered	less skewed	small	7.8	8.0	6.9	1.9
			large	6.6	6.9	5.6	3.3
		more skewed	small	7.4	7.7	6.4	4.4
			large	8.6	8.9	7.7	4.3
0.10	independent	less skewed	small	8.7	8.3	7.3	0.7
			large	10.5	10.3	9.3	0.8
		more skewed	small	14.6	14.5	13.4	0.8
			large	15.8	15.7	14.6	0.8
	Staggered	less skewed	small	12.0	11.9	11.3	1.2
			large	13.8	13.8	13.2	2.0
		more skewed	small	17.6	17.6	17.0	2.4
			large	19.0	19.0	18.4	2.9
0.01	independent	less skewed	small	11.5	9.8	10.8	1.0
			large	14.8	13.0	14.2	1.1
		more skewed	small	16.0	14.5	15.6	1.3
			large	15.3	13.9	14.9	1.3
	Staggered	less skewed	small	19.0	18.4	18.8	1.1
			large	23.4	22.8	23.3	2.1
		more skewed	small	25.0	24.5	25.1	3.6
			large	24.3	23.7	24.3	3.8

As the sample size was increased from 50 to 75 (Table 15), bias became consistently smaller for the MLE method than for the regression methods for all but the 50^{th} percentile. It was also interesting to note that bias for the regression methods actually became larger as the sample size increased while it became smaller for the MLE method.

5.4 Genetic Algorithm results

SAS MACRO code for the genetic algorithm was successfully developed (Appendix 3). The code involved nested MACRO subroutines, character conversion to numbers, and conversion of data sets into MACRO inputs.

The genetic algorithm was applied successfully on a Linux server with 4 Xeon processors running the Linux 2.6.5 kernel and SAS version 9.0. It was also applied successfully on an Optiplex GX260 and GX270 Dell computer running windows XP with SAS version 9.0. Run times varied from 20 minutes to more than 2 hours.

For a less-skewed small-variance distribution, the genetic algorithm performed very well according to the ratio of improvement criterion as all of the runs had an improvement ratio of .9 or greater (Table 16). The final mese value itself was very small for nearly all the runs when compared to simulated values of similar distributions (Tables 12 and 13) as a reference. It is interesting to note that the mese was reduced dramatically when either G or M was increased from 5 to 50. Generally, the trend is that the increase of M from 5 to 50 is more effective than increasing G from 5 to 50. Thus, it appears that having a larger number of designs to cross over and mutate is more effective in finding a near optimal design than having more generations.

Table 16: Genetic Algorithm results for alpha=63, Beta=7 and estimating the 50th percentile

Input				Output							Performance	
N	S	G	M	n_0	t_0	C	$?_1$	$?_2$	$?_3$	a	mese	ratio **
50	100	5	5	1	28	5	38	19	4	0.48	5.09	0.921
			50	2	29	2	41	12	1	0.51	0.05	0.999
		50	5	2	35	3	49	23	6	0.01	0.02	1.000
			50	2	29	5	29	13	4	0.13	0.04	0.999
	1000	5	5	2	39	4	47	7	5	0.11	0.21	1.000
			50	2	38	5	46	4	4	0.35	0.00	1.000
		50	5	3	19	0	52	1	1	0.20	0.05	0.999
			50	2	29	5	29	13	4	0.13	0.04	0.999
		Reference***		3	7	1	14	7	3	0.5	6.20	
		Reference***		7	7	0	7	7	7	NA	33.7	
75	100	5	5	3	29	4	52	16	3	0.09	1.06	0.976
			50	3	37	1	50	11	1	0.17	0.01	1.000
		50	5	1	38	2	46	10	4	0.31	1.67	0.983
			50	1	29	4	51	16	1	0.28	0.29	0.996
	1000	5	5	4	28	1	44	12	1	0.67	0.10	0.999
			50	1	40	1	33	8	3	0.21	0.01	1.000
		50	5	3	41	0	52	19	2	0.54	4.91	0.944
			50	2	29	0	42	12	3	0.36	0.05	0.999
		Reference***		3	7	1	14	7	3	0.5	3.00	
		Reference***		7	7	0	7	7	7	NA	2.4	

*Scenario failed to function because of floating point errors

**Ratio=1- $\frac{\text{Best mese}_{\text{Generation G}}}{\text{Average mese}_{\text{Generation 0}}}$

***Design used in simulation study—see tables 12 and 13

Some design parameter values that the genetic algorithm produced seem large, especially $?_1$. Nevertheless, the low values for *a* indicate that the sampling design will almost immediately switch to $?_3$,which is very small. Thus, the sampling design brings the user immediately close to the median, then slows to very small sampling increments around the hypothesized parameter of interest. Surprisingly, the larger sample size of 75 did not always produce designs with smaller mese values.

The efficiency of the algorithm was high in early generations (Figure 4). After 5 to 10 generations there was little improvement in the mese values. The one glaring exception occurred with G=50, M=5, S=1000 and N=75. The initial five designs all had large mese values. The crossover and mutation subroutines slowly improved the mese values but in the later generations the probability of mutation was so small that further improvement was unlikely. This illustrates the danger in having a small M, regardless of G. A different mutation tuning constant may also have had a positive effect.

S, the number of simulation iterations, appeared to have little effect on the efficiency of the eventual outcome of the genetic algorithm. Nevertheless, it must be kept in mind that the mese and the ratio values are more reliable when S is large.

Figure 4: Boxplots of various runs of the genetic algorithm progression of selected runs. a = 63 ß = 7

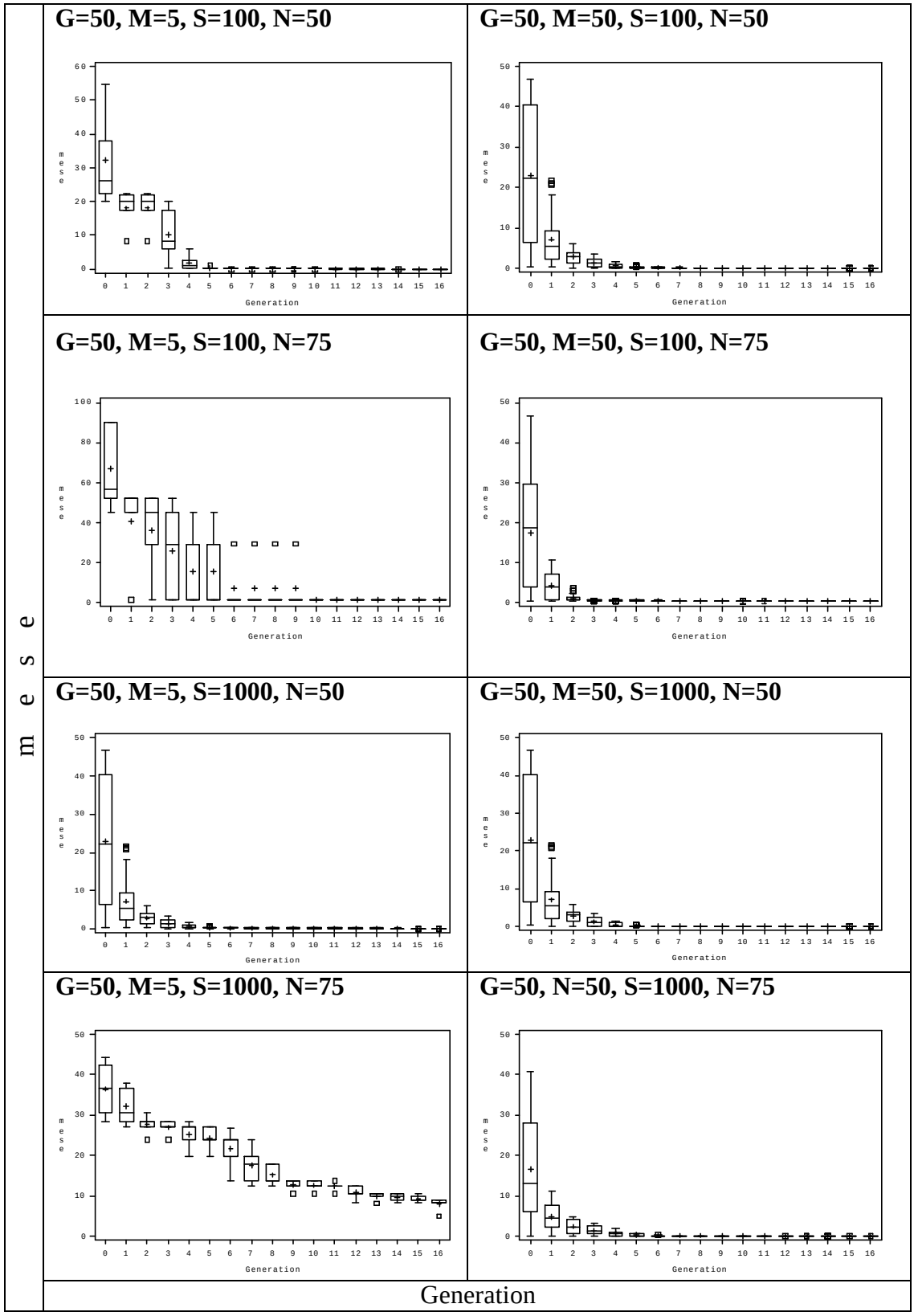


Table 17: Genetic Algorithm results for alpha=67, Beta=2 and estimating the 50th percentile

Input				Ouput							Perfomance	
N	S	G	M	n_0	t_0	C	$?_1$	$?_2$	$?_3$	a	mese	ratio **
50	100	5	5	1	14	4	27	9	1	0.06	18.19	0.967
			50	2	14	1	15	4	1	0.03	11.32	0.989
		50	5	5	12	3	26	9	1	0.13	12.99	0.990
			50	3	7	5	8	1	1	0.80	12.12	0.970
	1000	5	5	5	9	4	34	8	1	0.36	24.57	0.951
			50	*	*	*	*	*	*	*	*	*
		50	5	3	15	2	26	1	8	0.12	16.44	0.983
			50	*	*	*	*	*	*	*	*	*
		Reference***		3	7	1	14	7	3	0.5	65.1	
		Reference***		7	7	0	7	7	7	NA	96.8	
75	100	5	5	7	6	3	26	6	2	0.04	9.75	0.987
			50	8	10	4	21	11	1	0.02	7.81	0.989
		50	5	3	6	1	18	5	1	0.23	7.57	0.980
			50	*	*	*	*	*	*	*	*	*
	1000	5	5	3	7	5	8	1	1	0.75	12.12	0.974
			50	8	8	5	7	1	1	0.04	9.87	0.986
		50	5	6	15	0	30	3	11	0.04	9.45	0.990
			50	*	*	*	*	*	*	*	*	*
		Reference***		3	7	1	14	7	3	0.5	39.4	
		Reference***		7	7	0	7	7	7	NA	34.6	

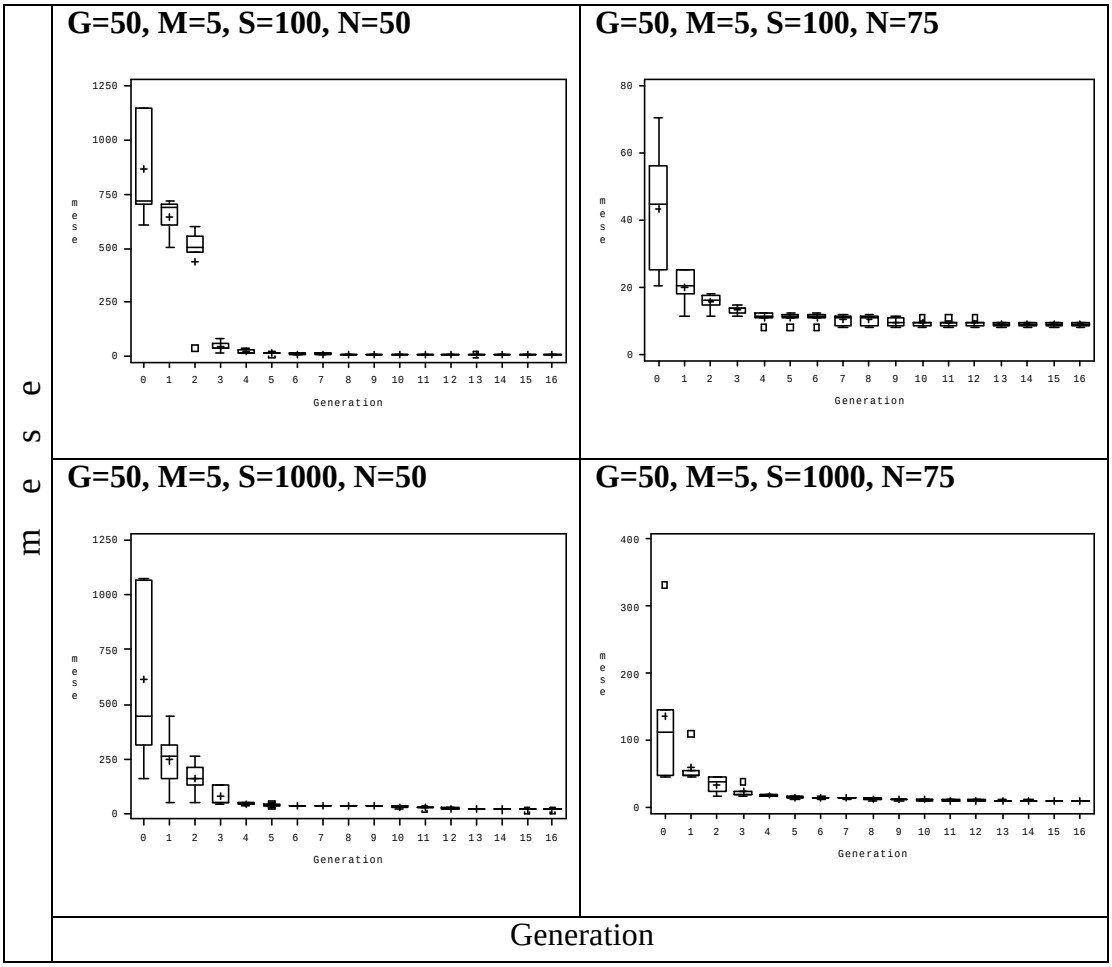
*Scenario failed to function because of floating point errors

**$\text{Ratio} = 1 - \dfrac{\text{Best mese}_{\text{Generation G}}}{\text{Average mese}_{\text{Generation 0}}}$

***Design used in simulation study—see Tables 12 and 13

Unfortunately, when S=1000 the genetic algorithm often terminates due to computational errors that occur in PROC LIFEREG. This happens more often when the hypothesized distribution is more skewed (Table 17). Nevertheless, when successful, the genetic algorithm converges quickly for both distributions (Figures 4 and 5).

Figure 5: Boxplots of the generations of various runs of the genetic algorithm where a = 67 ß = 2.



Not surprisingly, for the more-skewed large-variance distribution the genetic algorithm produced designs with larger mese values (Table 17). The design parameters reflect the differences in the two distributions. With the less skewed distribution, t_0 and $?_1$ were quite large, reflecting the lightness of the left tail. The more skewed distribution had smaller values for t_0 and $?_1$, reflecting the heavier left tail.

Chapter 6

Conclusions

This chapter discusses the results and their implications as well as ideas for future research.

6.1 Simulation Study

The simulation study has shown that as compared to the regression methods, the MLE method has higher coverage, better asymptotic properties in regards to bias, and lower mese values, especially when shelf life is defined by smaller percentiles. The MLE method still has problems, as demonstrated by coverage values less than the target when the sample size is small. Also, in practice, staggered designs are often used rather than truly independent designs. In such cases, the MLE method is only approximately valid. Nevertheless, this project clearly demonstrated that the MLE techniques are superior in every way to the regression methods and thus should be used.

Occasionally, the regression techniques were relatively unbiased and had relatively high coverage for the 50^{th} percentile (Table 11). If the standard errors were correctly calculated through a method such a bootstrapping perhaps the techniques would be useful for estimating the median.

The question remains why the MLE techniques, which have a sound theoretical basis, would ever perform worse than the regression techniques. It appears that bias due to censoring is sometimes balanced out by the bias that may be inherent in staggered designs. The standard errors of the regression techniques are extremely small and thus, occasionally but unpredictably, the regression techniques produce values consistently close to the true median. This seems to occur by chance, as demonstrated by the unexpected but consistent change in the bias of the fiftieth percentile for the sample size of 75 versus 50.

6.2 Genetic Algorithm

The genetic algorithm was successfully programmed and performed well according to the improvement ratio. The designs produced by the algorithm almost always had mese values that were smaller than the staggered design that is used commonly in practice, as shown by the simulation study. The generated designs are very different from the designs used in practice in almost all the parameter values. The genetic algorithm should thus be used to plan future shelf life studies.

The genetic algorithm performs best when M is large. The size of G seems to be secondary, and because of computational errors in PROC LIFEREG the iterations (S) should remain small.

6.3 Suggestions for future research

Only destructive sampling in the simulation and genetic algorithm was examined. This decision was because of the computing errors in PROC LIFEREG where 'floating point errors' frequently occyr. These occurred when interval censoring was involved and shut the program down. These 'floating point errors' also occurred in destructive sampling but only very rarely. To study the effects of non-destructive sampling on the properties of the estimators, another program (such as PROC NLP) must be employed to obtain the maximum likelihood estimates.

Another area of interest would be to vary the possible values (Table 9) of the parameters in the genetic algorithm. Other choices may lead to stronger and more practical designs. Additionally, the arguments for the Beta distribution in the mutation subroutine were arbitrary. Other arguments may improve the genetic algorithm. Similarly, the tuning constant for the mutation algorithm was arbitrarily chosen to be 0.10. Using this value, by generation 25, there is only an 8% chance of mutating any parameter. Perhaps a smaller tuning constant allowing mutation in later generations would be effective in some cases.

Work has been done on the design and analysis of staggered or adaptive sampling in other applications besides food shelf life studies. The Bruceton method is used in explosives research, (Wild and Collani, 2002) and a similar method is used in studying hearing loss (Gajewski, Sedwick, and Antonelli 2004). Further work is needed in applying this research in food shelf life estimation, and in developing a likelihood that takes into account the non-independent nature of the data due to staggered sampling.

Bayesian methods may be useful in overcoming small sample problems and achieving close to the target 95% coverage with data from staggered and independent designs.

Appendix 1: SAS code for Coffee Data

```
options FORMDLIM="#";

data d1;
infile datalines missover;
input date $ week a1-a30;
drop date--a30;
array a(30) a1-a30;
do i=1 to 30;
if a(i)=1 then do;
fail=0; begin=week; end=.; output;
end;
else if a(i)=0 then do;
fail=1; begin=.; end=week; output;
end;
end;
datalines;
250Jun004        0       1       1       1


50Sep004 15      1       1       1       1


210Dec004        26      0       1       1       1       0


40Feb005 32.5    0       1       0       1       1       0


220Mar005        39      0       1       0       1       1       1       1       1       0
        0

120Apr005        42      1       0       1       0       0       1       0       0       1
        0       0       0       0       0       1

20May005 45      0       0       0       0       0       0       0       0       1       0
        1       1       0       1       1       1       1       0       0       0       0
        1       0       0       1       1
proc print data=d1;
run;
proc lifereg data=d1;
model (begin,end)=;
output out=d2 quantiles=.01 .1 .25 .5 std=std p=predtime;
run;
proc print data=d2;
run;
data gac1;
set d1;
if fail=1;
retain rnk 0;
if fail=1 then rnk=rnk+1;
time=end;
ltime=log(time);
proc rank data=gac1 out=gac2 descending;
var rnk;
ranks rank;
proc print data=gac2;
run;
data gac3;
set gac2;
haz=100/rank;
retain chaz 0;
chaz=chaz+haz;
lchaz=log(chaz);
lchazc=lchaz;
if chaz gt 100 then lchazc=.;


run;
proc print data=gac3;
run;
data predict;
input lchaz lchazc;
cards;
.002180575 .002180575
 1.02268  1.02268
 1.45891  1.45891
 1.84083  1.84083
;
run;
data predict1;
set  gac3 predict;
```

```
run;
proc print data=predict1;
run;
proc print data=predict;
run;
proc print data=gac3;
run;
proc plot data=gac3;
plot ltime*lchaz;
plot ltime*lchazc;
run;
proc reg data=predict1 outest=gac4 covout;
/*model ltime=lchaz/r cli clm;*/
lrec: model lchaz=ltime/r cli clm;
output out=se p=pred stdp=see;
run;
proc print data=se;
run;
proc print data=gac4;
run;
data realse;
set se:
realse=(10**pred*see)*log(10);
pred1=10**pred;
run;

proc print data=realse;
run;
proc reg data=gac3 outest=gac4;
gac: model ltime=lchaz;
lrec: model lchaz=ltime;
run;

proc print data=gac4;
run;
data first;
set gac4:
if _model_='gac' then shelf=10**(intercept+lchaz*log10(100*(-log(.99))));
if _model_='lrec' then shelf=10**((log10(100*(-log(.99)))-intercept)/ltime);
run;
data five;
set gac4:
if  model ='gac' then shelf=10**(intercept+lchaz*log10(100*(-log(.95))));
if _model_='lrec' then shelf=10**((log10(100*(-log(.95)))-intercept)/ltime);
run;
data tenth;
set gac4:
if  model ='gac' then shelf=10**(intercept+lchaz*log10(100*(-log(.9))));
if _model_='lrec' then shelf=10**((log10(100*(-log(.9)))-intercept)/ltime);
run;
data twenty-five;
set gac4:
if _model_='gac' then shelf=10**(intercept+lchaz*log10(100*(-log(.75))));
if _model_='lrec' then shelf=10**((log10(100*(-log(.75)))-intercept)/ltime);
run;
data fifty;
set gac4:
if _model_='gac' then shelf=10**(intercept+lchaz*log10(100*(-log(.5))));
if _model_='lrec' then shelf=10**((log10(100*(-log(.5)))-intercept)/ltime);
run;
proc print data=first;
run;
proc print data=five;
run;
proc print data=tenth;
run;
proc print data=twenty-five;
run;
proc print data=fifty;
run;
proc reg data=predict1 outest=gac6;
labuz: model ltime=lchaz/r cli clm;
/*fun: model lchazc=ltime;*/
output out=se p=pred stdp=see;
run;
proc print data=se;
run;
data realse1;
set se:
realse=(10**pred*see)*log(10);
pred1=10**pred;
```

```
run;
proc print data=realse1;
run;
proc reg data=gac3 outest=gac6;
labuz: model ltime=lchazc;
fun: model lchazc=ltime;
run;
data first;
set gac6;
if _model_='labuz' then shelf=10**(intercept+lchazc*log10(100*(-log(.99))));
run;
data tenth;
set gac6;
if _model_='labuz' then shelf=10**(intercept+lchazc*log10(100*(-log(.9))));
run;
data twentyfive;
set gac6;
if  model_='labuz' then shelf=10**(intercept+lchazc*log10(100*(-log(.75))));
run;
data median;
set gac6;
if  model_='labuz' then shelf=10**(intercept+lchazc*log10(100*(-log(.5))));
run;
proc print data=first;
run;
proc print data=tenth;
run;
proc print data=twentyfive;
run;
proc print data=median;
run;

*this is for the taro;
data Aram1;
input ltime lhaz;
cards;
1.414973348      0.721246399
1.414973348      1.034175618
1.511883361      1.222744202
1.511883361      1.360802869
1.511883361      1.471551815
1.591064607      1.565382475
1.591064607      1.647900078
1.591064607      1.722520656
1.591064607      1.791530234
1.62324929       1.85659079
1.62324929       1.919016614
1.62324929       1.979949941
1.62324929       2.040499354
1.62324929       2.101886659
1.62324929       2.165661596
1.62324929       2.23411376
1.62324929       2.311274741
1.62324929       2.406155047
1.62324929       2.549951743
. 0.00218
. 1.02268
. 1.45891
. 1.84083
proc reg data=Aram1 outest=gac8;
Aram1: model  ltime=lhaz;
output out=se p=pred stdp=see;
run;

proc print data=se;
run;
data realse;
set se;
realse=(10**(pred)*see)*log(10);
pred1=10**pred;
run;
proc print data=realse;
run;
*method we are not looking at;

/*proc reg data=Aram outest=gac9;
Aram2: model ltime1=lhaz;
run;
data gac11;
set gac9;
if _model_ ='Aram2' then shelf=10**(intercept+lhaz*log10(69.3));
```

```
run;*/
data first;
set qac8;
if  model_='Aram1' then shelf=10**(intercept+lhaz*log10(100*(-log(.99))));
run;
data tenth;
set qac8;
if  model_='Aram1' then shelf=10**(intercept+lhaz*log10(100*(-log(.9))));
run;
data twentyfive;
set qac8;
if _model_='Aram1' then shelf=10**(intercept+lhaz*log10(100*(-log(.75))));
run;
data median;
set qac8;
if  model_='Aram1' then shelf=10**(intercept+lhaz*log10(100*(-log(.5))));
run;

proc print data=first;
run;
proc print data=tenth;
run;
proc print data=twentyfive;
run;
proc print data=median;
run;
```

Appendix 2: SAS code for the Simulation

```
options ls=60;
options FORMDLIM="#";

data nameG;
input Technique$;
datalines;
Gacula
:
run;
data nameMLE;
input Technique$;
datalines;
MLE
:
run;
data nameSloan;
input Technique$;
datalines;
Sloan
:
run;
data nameLabuza;
input Technique$;
datalines;
Labuza
;
run;
%macro Sim (iter= ,alpha= ,beta= ,nsamp= ,ntest= ,sttime= ,int1= ,int2= ,int3= ,stsize= ,c=
,accelerate= );

proc iml;
/**  define parameters;*/

/**  nit is the number of iterations;*/
nit= &iter;


/**setting up the real Weibull distribution;*/
alpha= &alpha ;
/*63.14;*/
beta= &beta;
/**7.17;*/
/*alpha=50;
beta=1.2;
*/
/**find the median of the Weibull;*/
med=alpha*((-log(.5))**(1/beta));
first=alpha*((-log(.99))**(1/beta));
tenth=alpha*((-log(.9))**(1/beta));
twentyfifth=alpha*((-log(.75))**(1/beta));
fiftieth=alpha*((-log(.5))**(1/beta));
print fiftieth;


/** for use in detecting coverage;*/
firstcol=j(nit,1,first);
tenthcol=j(nit,1,tenth);
twentyfifthcol=j(nit,1,twentyfifth);
fiftiethcol=j(nit,1,fiftieth);


/**setting up the total # we will ever sample;*/
nsamp=&nsamp;
/**25;*/

/**how many times we can test before censored;*/
ntest=&ntest;

/**when the first sample is taken;*/
sttime=&sttime;
/**14;*/
time=sttime;

/**for a loop later on about when to begin accelarated testing;*/
half=0;

/**How many days between samples before first failure;*/
int1= &int1;
/**14;*/

/**# of the days between samples after first failure;*/
```

```
int2= &int2;
/*7*/

/**# of days between samples after half the samples have failed;*/
int3= &int3;
/*3*/

/**random #;*/
seed=1;

/**n for first sample at sttime;*/
stsize= &stsize;
/**currentsize=stsize;*/
dumsize=stsize;
   time=sttime;

sint1=&c;
sint2=&c;
sint3=&c;
a=&accelerate;

/****  does this do anything?  we just reset currentsize before we do anything anyway;*/

/**  start iterations;*/
do i=1 to nit;

/****data=j(nsamp,5,-9);*/
   data=j(nsamp,7,-9);
/*   * so that we can put iteration number in the array for later use in a by statement;*/
   cfail=0;
   tfail=0;

dumsize=stsize;
time=sttime;
/**for coverage later;*/
cover1=j(4,1,.);
c= {.002180575 .002180575  .002180575,
 1.02268  1.02268 1.02268,
 1.45891  1.45891 1.45891,
 1.84083  1.84083 1.84083 };
 do j=1 to 4;
 cover1[j,1]=i;
 end;
 cover2=c||cover1;


;
/**  generate data:*/
   do j=1 to nsamp;
      data[j,1]=rand('weibull', beta, alpha);  %*** the actual failure time of the unit;
           data[j,2]=0;                            %*** number of times the unit has been tested;
           data[j,3]=0;                            %*** contains the censoring variable (0,1,2);

                                                                                                %***
column 4 the observed failure or censoring time of the unit;

*     data[j,5]=ranuni(0);                    %*** random numbers for random sorting and sampling of
the nonfailed units units;
    data[j,6]=.;                             %*** for storage of next to last observed value;
        data[j,7]=i;                                                    %*** iteration number;
        end;

   done=0;

   do until (done>1);


/** print currentsize time data;*/

      do j=1 to nsamp;
             data[j,5]=ranuni(seed);
          end;

/**assigns code so that censored and failed units will move to the end of the array;*/
      do j=1 to nsamp;
         if data[j,3]>0
         then data[j,5]=2;
      end;

/**** print currentsize time data;*/
```

```
/**sorts data by the random #'s for sampling;*/
      scdata=rank(data[,5]);
      dum=data;
      do w= 1 to 6;
         data[scdata,w]=dum[,w];
      end;

/**** print currentsize time data;*/

sfail=0;

/**stops resampling failed or censored units and samples;*/
      currentsize=dumsize;
if currentsize>nsamp
then currentsize=nsamp;
      do j=1 to currentsize;

/****I think sfail is the number of failures in this sample, cfail is the cumulative number of failues
in this study,*/
/*                and tfail is an indicator of whether there were multiple failures in this sample;*/

         if data[j,3]>0
           then goto here;
           else
              data[j,2]=data[j,2]+1;


         if data[j,1]<time
               then
               do:
                  data[j,4]=time;
                  data[j,3]=1;
                  cfail=cfail+1;
                  sfail=sfail+1;
              if sfail>1
                then tfail=1;
              end;
              else
                     if data[j,2]=ntest
                   then
                         do:
                           data[j,4]=time;
                           data[j,3]=2;

                                    end;
/*               *this sets up the beginning part of the interval censored data when the sampling is
not destructive:*/
                           if data[j,2]<ntest & data[j,4]<data[j,1]
                           then
                             do:
                                  data[j,6]=time;
                                  end;




         here:    ;



      end;

***print currentsize time data;
/*print done sfail cfail tfail dumsize currentsize time half;*/
if cfail=0 then goto there;


if sfail >= (currentsize*a) & half=0
        then do;

                half=1;

                end;

if  half=0
        then do;
        time=time+int2;
                dumsize=currentsize+sint2;
                end:
if half=0 then goto hither;
```

```
if half=1
        then do;
                time=time+int3;
                half=1;
                dumsize=currentsize+sfail+sint3;
                end;
hither: ;
if cfail ^=0 then goto thither;
 there:        do;
         time=time+int1;
                  dumsize=currentsize+sint1;
              end;
thither: ;
/*print done sfail cfail tfail dumsize currentsize time half;*/
/**current group:*/
   done=min(data[,5]);



   end;

/*print half time dumsize sfail cfail currentsize data;*/
rankdata=1+nsamp-rank(data[,4]);
hazdata=j(nsamp,1);
sumhaz=0;
cumhaz=j(nsamp,1);
do ijk=1 to nsamp;

if data[ijk,3]=1
then
hazdata[ijk]=1/rankdata[ijk];
else hazdata[ijk]=.;

/**Newdata=rankdata||hazdata||(data[,4])||(data[,1]);*/
/**varnames='rank'//'hazard'//'observed'//'real';*/
/**print varnames;*/

end:
masterdat=rankdata||hazdata||(data[,4])||(data[,1]);
     call sort( masterdat, {1 4}, {1} );
do ijk=1 to nsamp;
 if masterdat[ijk,2] ^= .
 then do;
cumhaz[ijk]=sumhaz+masterdat[ijk,2];
sumhaz=cumhaz[ijk]; end;
 else cumhaz[ijk]=.;
end;
/*print masterdat cumhaz;*/
/**Plotting method;*/
x1=(cumhaz)*100;
y1=(masterdat[,3]);
X=x1||y1||(1+nsamp-rank(masterdat[,3]));

/**Setting up the interval, right, and left censored nature of data with two columns, begin and end*/
/*right censored is begin=time end=infinity, left censored begin=. or 0, interval censored occurs
when*/
/*ntest>1 or the sampling is not destructive, then we have an interval where we know the food was
good*/
/*till the food was bad.;*/

begin=j(nsamp,1,.);
end=j(nsamp,1,.);
del=j(nsamp,1,.);
sumdel=0;
cumdel=j(nsamp,1,.);
do ijk=1 to nsamp;
 if data[ijk,3] = 1
  then begin[ijk]= data[ijk,6];
   else begin[ijk]=data[ijk,4];
 if data[ijk,3] = 1
  then end[ijk]=data[ijk,4];
   else end[ijk]=.;
end;

do ijk=1 to nsamp;
 if end[ijk]^=.
   then del[ijk]=1;
end;

do ijk=1 to nsamp;
if end[ijk]^=.
```

```
then do;
cumdel[ijk]=sumdel+del[ijk];
 sumdel=cumdel[ijk]; end;
 else cumdel[ijk]=.;
end:
maxdel=j(nsamp,1,.);

do ijk=1 to nsamp;
  maxdel[ijk]=max(cumdel);
end:
  cmaxdel=j(4,1,.);

do ijk=1 to 4;
 cmaxdel[ijk]=max(cumdel);
end;

/**print begin end data ;*/

labuza=j(nsamp,5,.);
labhaz=j(nsamp,1,.);
laby=j(nsamp,1,.);
iter=j(nsamp,1,i);
Control=j(nsamp,1,.);

do ijk=1 to nsamp;/*change in real simulation to nsamp*/
if X[ijk,1] <100
then
labuza[ijk,1]=X[ijk,1];
end:
do ijk=1 to nsamp;
if x[ijk,1] <100 & x[ijk,1]^=.
then
labuza[ijk,2]=X[ijk,2];
labuza[ijk,3]=X[ijk,3];
labhaz[ijk]=labuza[ijk,1];
laby[ijk]=labuza[ijk,2];
/*if x[ijk,1]>100 & x[ijk,1]^=.*/
/*then*/
/*labuza[ijk,4]=X[ijk,3];*/

if labuza[ijk,1]=.
then labuza[ijk,3]=.;
end;

number=1;
S5=j(nsamp,1,.);
do ijk=1 to nsamp;
labuza[ijk,5]=number;
S5[ijk,1]=number;
number=number+1;
end:
Sloan2=j(nsamp,1,.);
Sloan1=(labuza[,2])||(labuza[,5])||Sloan2;

call sort (Sloan1,{2},{2});

Sloan1=Sloan1||S5;


dead=0;

do ijk=1 to nsamp;

if Sloan1[ijk,1] ^= .
then do;
dead=1;
end:
if dead=1 then goto rest;

if Sloan1[ijk,1] = .
then do; Sloan1[ijk,3]=Sloan1[ijk,4];
end;
if dead=0 then goto active;

rest:
dead=1;
active: ;
end;

Sloanrank=max(Sloan1[,3]);
```

```
do ijk=1 to nsamp;
if labuza[ijk,3] ^=.
then do;
labuza[ijk,3]=labuza[ijk,3]-Sloanrank;
end;
end:
Sloan=(labuza[,2])||(labuza[,3]);


Shazdata=j(nsamp,1);
Ssumhaz=0;
Scumhaz=j(nsamp,1);
do ijk=1 to nsamp;

if Sloan[ijk,1] ^= .
then
Shazdata[ijk]=1/Sloan[ijk,2];
else Shazdata[ijk]=.;
end;
Sloan=Sloan||Shazdata;
do ijk=1 to nsamp;
if Sloan[ijk,1] ^= .
then do;
Scumhaz[ijk]=Ssumhaz+Sloan[ijk,3];
Ssumhaz=Scumhaz[ijk]; end;
else Scumhaz[ijk]=.;
end;


/** Sloan method;*/

Sx1=(Scumhaz)*100;
Sy1=(Sloan[,1]);

lSx1=log10(Sx1);
lSy1=log10(Sy1);




lx1=log10(x1);
ly1=log10(y1);
loglabhaz=log10(labhaz);
loglaby=log10(laby);


Out=lx1||ly1||iter||loglabhaz||loglaby||lSx1||lSy1||maxdel;

Perc=first||tenth||twentvfifth||fiftieth;
Life=begin||end||iter||data[,1]||maxdel;
coverage=firstcol||tenthcol||twentyfifthcol||fiftiethcol;

cover2a=cover2||cmaxdel;

/*print data Life;*/

if nit=1 then Life1=Life;
else Life1=Life1//Life;

if nit=1 then Out1=Out;
else Out1=Out1//Out;

 if nit=1 then cover3=cover2a;
 else cover3=cover3//cover2a;

end;
create cover from coverage;
append from coverage;

create med1 from Out1;
append from Out1;

create Reg from Life1;
append from Life1;


create Percentile from Perc;
append from Perc;
```

```
create linear from cover3;
append from cover3;
quit;



data Lifereg (rename= (COL1=begin COL2=end COL3=iter COL4=real COL5=del));
set Reg;
control=0;
retain dum 0;
if COL3 ne dum then control=1;
dum=COL3;
if COL2=1000 then COL2=.;
if COL5<3 then delete;
run;

ods listing close;
proc univariate data=reg;
var COL4;
run;
ods listing;



proc lifereg data=Lifereg noprint; * inest=outest;
*where iter~=1;* & iter~=1495;*&iter~=186&iter~=184&iter~=183&iter~=1&iter~=182&iter~=180&iter~=181;
by iter;
model (begin, end)= /  maxiter=50  ;*intercept=4.09 initial=0 to 5 by .1; *intercept=4.09;
output out=d2 quantiles=.01 .1 .25 .5 std=std p=predtime control=control  ;
run;

data d2; set d2;
if std>1000 then delete;
run;




data lrfirst;
set d2;
 Model ='Lifereg';
if _PROB_ ^= .01
then delete;
first=predtime;
run;
data lrtenth;
set d2;
 Model ='Lifereg';
if _PROB_ ^= .1
then delete;
tenth=predtime;
run;

data lrtwentyfifth;
set d2;
 Model ='Lifereg';
if _PROB_ ^= .25
then delete;
twentyfifth=predtime;
run;

data lrfifty;
set d2;
 Model ='Lifereg';
if _PROB_ ^= .50
then delete;
fiftieth=predtime;
run;

data Percentile (rename= (COL1=first COL2=tenth COL3=twentyfifth COL4=fiftieth));
set Percentile;
run;
*begin coverages;



data LFI;
      set d2;
      ltime=(predtime);
      *stde=std/predtime;
           lower=(ltime-1.96*std);
```

```
      upper=(ltime+1.96*std);

   run;

data coverage (rename= (COL1=first COL2=tenth COL3=twentyfive COL4=fifty));
set cover;
run;


*covers fiftieth percentile;
data c50A;
set LFI;
if _PROB_ ^=.50 then delete;
run;
data c50B;
set coverage;
keep fifty;
run;
/*proc print data=c50B;*/
/*run;*/
data c50C;
merge c50A c50B;
if std =. then delete;
if lower<fifty<upper then coverage=1;
else coverage=0;
run;



ODS output  Moments=c50Da;
ods listing close;

proc univariate data=c50C;
var coverage;
run;
ods listing;

data c50E (rename= (cValue1=MLEcoveragefifty));
set c50Da;
if Label1 ^= 'Mean' then delete;
run;

data c50F;
set c50E;
keep MLEcoveragefifty;
run;




*covers twentyfifth percentile;
data c25A;
set LFI;
if _PROB_ ^=.25 then delete;
run;
data c25B;
set coverage;
keep twentyfive;
run;

data c25C;
merge c25A c25B;
if std =. then delete;
if lower<twentyfive<upper then coverage=1;
else coverage=0;
run;

ODS output  Moments=c25D;
ods listing close;
proc univariate data=c25C;
var coverage;
run;
ODS trace off;
ods listing;

data c25E (rename= (cValue1=MLEcoveragetwentyfive));
set c25D;
if Label1 ^= 'Mean' then delete;
run;
data c25F;
```

```
set c25E;
keep MLEcoveragetwentyfive;
run;


*coverage tenth percentile;
data c10A;
set LFI;
if _PROB_ ^=.10 then delete;
run;
data c10B;
set coverage;
keep tenth;
run;

data c10C;
merge c10A c10B;
if std =. then delete;
if lower<tenth<upper then coverage=1;
else coverage=0;
run;


ODS output  Moments=c10D;
ods listing close;
proc univariate data=c10C;
var coverage;
run;
ods listing;

data c10E (rename= (cValue1=MLEcoveragetenth));
set c10D;
if Label1 ^= 'Mean' then delete;
run;
data c10F;
set c10E;
keep MLEcoveragetenth;
run;




*coverage first percentile;
data c1A;
set LFI;
if _PROB_ ^=.01 then delete;
run;
data c1B;
set coverage;
keep first;
run;

data c1C;
merge c1A c1B;
if std =. then delete;
if lower<first<upper then coverage=1;
else coverage=0;
run;


ODS output  Moments=c1D;
ods listing close;
proc univariate data=c1C;
var coverage;
run;
ods listing;
ODS trace off;
data c1E (rename= (cValue1=MLEcoverageFirst));
set c1D;
if Label1 ^= 'Mean' then delete;
run;
data c1F;
set c1E;
keep MLEcoverageFirst;
run;


data coverageLF (rename=( MLEcoveragefifty=Fiftieth  MLEcoveragetwentyFive=TwentyFifth
MLEcoverageTenth=Tenth  MLEcoverageFirst=First)); ;
merge nameMLE c50F c25F c10F  c1F  ;
```

```
run;


data first;
set Percentile;
_REF_=first;
keep _REF_ _REFLAB_;
_REFLAB_='First Percentile';
keep _REF_ _REFLAB_;
run;
data tenth;
set Percentile;
 REF =tenth;
 REFLAB ='Tenth Percentile';
keep _REF_ _REFLAB_;
run;
data twentyfifth;
set Percentile;
_REF_=twentyfifth;
_REFLAB_='twentyfifth Perc';
keep _REF_ _REFLAB_;
run;
data fiftieth;
set Percentile;
_REF_=fiftieth;
_REFLAB_='Fiftieth Percentile';
keep _REF_ _REFLAB_;
run;

data estimate(rename=(COL1=GacX COL2=GacY COL3=Iteration COL4=LabX COL5=Laby COL6=SloanX COL7=SloanY
COL8=del));
set med1:
if COL8<3 then delete;
run;


data estimate1 :
set estimate(Firstobs=1 obs=1);
_REF_=med;
keep _REF_;
run;

data linear1 (rename= (COL1=SloanX COL2=GacX COL3=LabX COL4=Iteration COL5=del));
set linear;

if 0.5>COL2>0.001
then coverage=.01;

if 1.1>COL2>.04
then coverage=.1;

if 1.5>COL2>1.1
then coverage=.25;

if 2.0>COL2>1.5
then coverage=.5;

if COL5<3 then delete;
*drop SloanX LabX;
run;


/*proc print data=linear;*/
/*run;*/
data estimate1;
set estimate;
coverage=0;
run;

data predict;
set  estimate1 linear1;
run;

proc sort data=predict;
by iteration;
run;

proc reg data=predict noprint;
by Iteration;
Gaculapred: model GacY=GacX;
```

```
output out=GacCoverage p=pred stdp=error;
run;
proc reg data=predict noprint;
by Iteration;
Labuzapred: model LabY=LabX;
output out=LabCoverage p=pred stdp=error;
run;

proc reg data=predict noprint;
by Iteration;
Sloanpred: model SloanY=SloanX;
output out=SloanCoverage p=pred stdp=error;
run;

data SloanCoverage1;
set SloanCoverage;
if coverage = 0 then delete;
run;



data SloanCoverage2;
set SloanCoverage1;
realse=(10**(pred)*error)*log(10);
pred1=10**(pred);
_PROB_=coverage;
drop coverage;
run;



data SloanCoverage3;
      set SloanCoverage2;
           lower=(pred1-1.96*realse);
      upper=(pred1+1.96*realse);

   run;


data coverage (rename= (COL1=first COL2=tenth COL3=twentyfive COL4=fifty));
set cover;
run;



*covers fiftieth percentile;
data Sloan50A;
set SloanCoverage3;
if _PROB_ ^=.50 then delete;
run;

data Sloan25A;
set SloanCoverage3;
if _PROB_ ^= .25 then delete;
run;


data Sloan10A;
set SloanCoverage3;
if _PROB_ ^= .10 then delete;
run;

data Sloan01A;
set SloanCoverage3;
if _PROB_ ^= .01 then delete;
run;


data Sloan50B;
set coverage;
keep fifty;
run;

data Sloan25B;
set coverage;
keep twentyfive;
run;
data Sloan10B;
set coverage;
```

```
keep tenth;
run;
data Sloan01B;
set coverage;
keep first;
run;

data Sloan50C;
merge Sloan50A Sloan50B;
if realse =. then delete;
if lower<fifty<upper then coverage=1;
else coverage=0;
run;
data Sloan25C;
merge Sloan25A Sloan25B;
if realse =. then delete;
if lower<twentyfive<upper then coverage=1;
else coverage=0;
run;


data Sloan10C;
merge Sloan10A Sloan10B;
if realse =. then delete;
if lower<tenth<upper then coverage=1;
else coverage=0;
run;
data Sloan01C;
merge Sloan01A Sloan01B;
if realse =. then delete;
if lower<first<upper then coverage=1;
else coverage=0;
run;


ODS output  Moments=Sloan50Da;
ods listing close;

proc univariate data=Sloan50C;
var coverage;
run;
ods listing;

ODS output  Moments=Sloan25Da;
ods listing close;

proc univariate data=Sloan25C;
var coverage;
run;

ods listing;

ODS output  Moments=Sloan10Da;
ods listing close;

proc univariate data=Sloan10C;
var coverage;
run;
ods listing;

ODS output  Moments=Sloan01Da;
ods listing close;

proc univariate data=Sloan01C;
var coverage;
run;
ods listing;

/*proc print data=Sloan50Da;*/
/*run;*/
data Sloan50Db (rename= (cValue1=SloancoverageFifty));
set Sloan50Da;
if Label1 ^= 'Mean' then delete;
run;
data Sloan25Db (rename= (cValue1=SloancoverageTwentyFive));
set Sloan25Da;
if Label1 ^= 'Mean' then delete;
run;
data Sloan10Db (rename= (cValue1=SloancoverageTenth));
set Sloan10Da;
if Label1 ^= 'Mean' then delete;
```

```
run;
data Sloan01Db (rename= (cValue1=SloancoverageFirst));
set Sloan01Da;
if Label1 ^= 'Mean' then delete;
run;

data Sloan50Dc;
set Sloan50Db;
keep SloancoverageFifty;
run;

data Sloan25Dc;
set Sloan25Db;
keep SloancoverageTwentyFive;
run;

data Sloan10Dc;
set Sloan10Db;
keep SloancoverageTenth;
run;
data Sloan01Dc;
set Sloan01Db;
keep SloancoverageFirst;
run;



data LabCoverage1;
set LabCoverage;
if coverage = 0 then delete;
run;


data LabCoverage2;
set LabCoverage1;
realse=(10**(pred)*error)*log(10);
pred1=10**(pred);
_PROB_=coverage;
drop coverage;
run;
data LabCoverage3;
      set LabCoverage2;
           lower=pred1-1.96*realse;
      upper=pred1+1.96*realse;

   run;



data coverage (rename= (COL1=first COL2=tenth COL3=twentyfive COL4=fifty));
set cover;
run;



*covers fiftieth percentile;
data Lab50A;
set LabCoverage3;
if _PROB_ ^=.50 then delete;
run;

data Lab25A;
set LabCoverage3;
if _PROB_ ^= .25 then delete;
run;

data Lab10A;
set LabCoverage3;
if _PROB_ ^= .10 then delete;
run;

data Lab01A;
set LabCoverage3;
if _PROB_ ^= .01 then delete;
run;


data Lab50B;
set coverage;
keep fifty;
run;
```

```
data Lab25B;
set coverage;
keep twentyfive;
run;
data Lab10B;
set coverage;
keep tenth;
run;
data Lab01B;
set coverage;
keep first;
run;

data Lab50C;
merge Lab50A Lab50B;
if realse =. then delete;
if lower<fifty<upper then coverage=1;
else coverage=0;
run;
data Lab25C;
merge Lab25A Lab25B;
if realse =. then delete;
if lower<twentyfive<upper then coverage=1;
else coverage=0;
run;
data Lab10C;
merge Lab10A Lab10B;
if realse =. then delete;
if lower<tenth<upper then coverage=1;
else coverage=0;
run;
data Lab01C;
merge Lab01A Lab01B;
if realse =. then delete;
if lower<first<upper then coverage=1;
else coverage=0;
run;

ODS output  Moments=Lab50Da;
ods listing close;

proc univariate data=Lab50C;
var coverage;
run;
ods listing;

ODS output  Moments=Lab25Da;
ods listing close;

proc univariate data=Lab25C;
var coverage;
run;

ods listing;

ODS output  Moments=Lab10Da;
ods listing close;

proc univariate data=Lab10C;
var coverage;
run;
ods listing;

ODS output  Moments=Lab01Da;
ods listing close;

proc univariate data=Lab01C;
var coverage;
run;
ods listing;

/*proc print data=Lab50Da;*/
/*run;*/
data Lab50Db (rename= (cValue1=LabcoverageFifty));
set Lab50Da;
if Label1 ^= 'Mean' then delete;
run;
data Lab25Db (rename= (cValue1=LabcoverageTwentyFive));
set Lab25Da;
if Label1 ^= 'Mean' then delete;
run;
```

```
data Lab10Db (rename= (cValue1=LabcoverageTenth));
set Lab10Da;
if Label1 ^= 'Mean' then delete;
run;
data Lab01Db (rename= (cValue1=LabcoverageFirst));
set Lab01Da;
if Label1 ^= 'Mean' then delete;
run;

data Lab50Dc;
set Lab50Db;
keep LabcoverageFifty;
run;

data Lab25Dc;
set Lab25Db;
keep LabcoverageTwentyFive;
run;


data Lab10Dc;
set Lab10Db;
keep LabcoverageTenth;
run;
data Lab01Dc;
set Lab01Db;
keep LabcoverageFirst;
run;









data GacCoverage1;
set GacCoverage;
if coverage = 0 then delete;
run;

data GacCoverage2;
set GacCoverage1;
realse=(10**(pred)*error)*log(10);
pred1=10**(pred);
 PROB =coverage;
drop coverage;
run;
data GacCoverage3;
      set GacCoverage2;
          lower=pred1-1.96*realse;
      upper=pred1+1.96*realse;

   run;



data coverage (rename= (COL1=first COL2=tenth COL3=twentyfive COL4=fifty));
set cover;
run;



*covers fiftieth percentile;
data Gc50A;
set GacCoverage3;
if _PROB_ ^=.50 then delete;
run;

data Gc25A;
set GacCoverage3;
if _PROB_ ^= .25 then delete;
run;

data Gc10A;
set GacCoverage3;
if _PROB_ ^= .10 then delete;
run;

data Gc01A;
set GacCoverage3;
```

```
if _PROB_ ^= .01 then delete;
run;


data Gc50B;
set coverage;
keep fifty;
run;

data Gc25B;
set coverage;
keep twentyfive;
run;
data Gc10B;
set coverage;
keep tenth;
run;
data Gc01B;
set coverage;
keep first;
run;

data Gc50C;
merge Gc50A Gc50B;
if realse =. then delete;
if lower<fifty<upper then coverage=1;
else coverage=0;
run;
data Gc25C;
merge Gc25A Gc25B;
if realse =. then delete;
if lower<twentyfive<upper then coverage=1;
else coverage=0;
run;
data Gc10C;
merge Gc10A Gc10B;
if realse =. then delete;
if lower<tenth<upper then coverage=1;
else coverage=0;
run;
data Gc01C;
merge Gc01A Gc01B;
if realse =. then delete;
if lower<first<upper then coverage=1;
else coverage=0;
run;


ODS output  Moments=Gc50Da;
ods listing close;

proc univariate data=Gc50C;
var coverage;
run;
ods listing;

ODS output  Moments=Gc25Da;
ods listing close;

proc univariate data=Gc25C;
var coverage;
run;

ods listing;

ODS output  Moments=Gc10Da;
ods listing close;

proc univariate data=Gc10C;
var coverage;
run;
ods listing;

ODS output  Moments=Gc01Da;
ods listing close;

proc univariate data=Gc01C;
var coverage;
run;
ods listing;
```

```
data Gc50Db (rename= (cValue1=GaccoverageFifty));
set Gc50Da;
if Label1 ^= 'Mean' then delete;
run;
data Gc25Db (rename= (cValue1=GaccoverageTwentyFive));
set Gc25Da;
if Label1 ^= 'Mean' then delete;
run;
data Gc10Db (rename= (cValue1=GaccoverageTenth));
set Gc10Da;
if Label1 ^= 'Mean' then delete;
run;
data Gc01Db (rename= (cValue1=GaccoverageFirst));
set Gc01Da;
if Label1 ^= 'Mean' then delete;
run;

data Gc50Dc;
set Gc50Db;
keep GaccoverageFifty;
run;

data Gc25Dc;
set Gc25Db;
keep GaccoverageTwentyFive;
run;

data Gc10Dc;
set Gc10Db;
keep GaccoverageTenth;
run;
data Gc01Dc;
set Gc01Db;
keep GaccoverageFirst;
run;


data GacCoverage (rename=(Gaccoveragefifty=Fiftieth GaccoveragetwentyFive=TwentyFifth
GaccoverageTenth=Tenth GaccoverageFirst=First));;
merge nameG Gc50Dc Gc25Dc Gc10Dc Gc01Dc;
run;


ods listing;


proc reg data=estimate outest=gacest noprint;
by Iteration;
Gacula: model GacY=GacX;
output out=test p=pred;
run;


proc reg data=estimate outest=Rect noprint;
by Iteration;
Rectification: model GacX=GacY;
run;
proc reg data=estimate outest=Sloan noprint;
by Iteration;
Sloan: model SloanY=SloanX;
run;
data Sloan;
set Sloan;
if _model_='Sloan' then first=10**(intercept+SloanX*log10(100*(-log(.99))));
/*if  model ='Labuza' then fifth=10**(intercept+LabX*log10(100*(-log(.95))));*/
if _model_='Sloan' then tenth=10**(intercept+SloanX*log10(100*(-log(.9))));
if _model_='Sloan' then twentyfifth=10**(intercept+SloanX*log10(100*(-log(.75))));
if _model_='Sloan' then fiftieth=10**(intercept+SloanX*log10(100*(-log(.5))));
run;
proc reg data=estimate outest=labest noprint ;
by Iteration;
Labuza: model Laby=LabX;
run;
data Lab;
set labest;
if _model_='Labuza' then first=10**(intercept+LabX*log10(100*(-log(.99))));
/*if  model ='Labuza' then fifth=10**(intercept+LabX*log10(100*(-log(.95))));*/
if  model ='Labuza' then tenth=10**(intercept+LabX*log10(100*(-log(.9))));
if  model ='Labuza' then twentyfifth=10**(intercept+LabX*log10(100*(-log(.75))));
if _model_='Labuza' then fiftieth=10**(intercept+LabX*log10(100*(-log(.5))));
run;
```

```
/*proc print data=Gac;*/
/*run;*/
data Gac;
set gacest;
if  model ='Gacula' then first=10**(intercept+GacX*log10(100*(-log(.99))));
if  model ='Gacula' then fifth=10**(intercept+GacX*log10(100*(-log(.95))));
if _model_='Gacula' then tenth=10**(intercept+GacX*log10(100*(-log(.9))));
if  model ='Gacula' then twentyfifth=10**(intercept+GacX*log10(100*(-log(.75))));
if _model_='Gacula' then fiftieth=10**(intercept+GacX*log10(100*(-log(.5))));

run;
ods listing close;
proc univariate data=Gac;
var fiftieth;
run;
ods listing;


data combinedfifty;
set Gac Lab Sloan lrfifty;
run;


data LRMLE1;
set LRMLE;
if Label1 ^= 'N' then delete;
keep cValue1;
run;


data lrfiftyclean;
set lrfiftv;
if fiftieth=. then delete;
run;

data lrtwentyfifthclean;
set lrtwentvfifth;
if twentyfifth=. then delete;
run;

data lrtenthclean;
set lrtenth;
if tenth=. then delete;
run;



data lrfirstclean;
set lrfirst;
if first=. then delete;
run;


proc IML;
use fiftieth;
read all var { REF_} into real;
use lrfiftvclean;
read all var {fiftieth} into fifty;


N=nrow(fiftv);
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
fiftvMSE=fiftvP||fiftv;
num=(fiftyMSE[,1]-fiftyMSE[,2]);

num2=num`*num;
den=N-1;
fiftyMSELRa=num2/den;
fiftvMSELR=sum(fiftyMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
fiftyMSSELR=median(num1);
create FiftyMLE from fiftyMSSELR[colname='Fifty' rowname='MLE'];
   append from fiftyMSSELR[ rowname='MLE'];
```

```
quit;
data FiftyMLEA (rename= (_LIT1017=technique));
set FiftyMLE;
run;



proc IML;
use twentyfifth;
read all var {_REF_} into real;
use lrtwentyfifthclean;
read all var {twentyfifth} into fifty;
N=nrow(fiftv):
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
twentvfifthMSE=fiftvP||fiftv:
num=(twentyfifthMSE[,1]-twentyfifthMSE[,2]);

num2=num`*num;
den=N-1;
twentyfifthMSELRa=num2/den;
twentvfifthMSELR=sum(twentyfifthMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
twentyfifthMSSELR=median(num1);
/*print twentyfifthMSELR twentyfifthMSSELR N;*/
create TwentvfifthMLE from twentvfifthMSSELR[colname='Twentyfifth' rowname='MLE'];
   append from twentyfifthMSSELR[ rowname='MLE'];
quit;

data TwentyfifthMLEA (rename= (_LIT1017=technique));
set TwentyfifthMLE;
run;


proc IML;
use tenth;
read all var { REF_} into real;
use lrtenthclean;
read all var {tenth} into fifty;
/**print fifty real;*/


N=nrow(fiftv);
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
tenthMSE=fiftvP||fiftv:
num=(tenthMSE[,1]-tenthMSE[,2]);

num2=num`*num;
den=N-1;
tenthMSELRa=num2/den;
tenthMSELR=sum(tenthMSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
tenthMSSELR=median(num1);
/*print tenthMSELR tenthMSSELR N;*/
create tenthMLE from TenthMSSELR[colname='Tenth' rowname='MLE'];
   append from tenthMSSELR[ rowname='MLE'];
quit;


data tenthMLEA (rename= (_LIT1017=technique));
set tenthMLE;
run;


proc IML;
use first;
read all var {_REF_} into real;
use lrfirstclean;
```

```
read all var {first} into fifty;


N=nrow(fiftv):
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
firstMSE=fiftvP||fiftv:
num=(firstMSE[,1]-firstMSE[,2]);

num2=num`*num;
den=N-1;
firstMSELRa=num2/den;
firstMSELR=sum(firstMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
firstMSSELR=median(num1);
create firstMLE from firstMSSELR[colname='First' rowname='MLE'];
   append from firstMSSELR[ rowname='MLE'];
quit;


data firstMLEA (rename= (_LIT1017=technique));
set firstMLE;
run;



proc IML;
use fiftieth;
read all var {_REF_} into real;
use Gac;
read all var {fiftieth} into fifty;



N=nrow(fiftv):
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
fiftvMSE=fiftvP||fiftv:
num=(fiftyMSE[,1]-fiftyMSE[,2]);

num2=num`*num;
den=N-1;
fiftvMSELRa=num2/den;
fiftvMSEGac=sum(fiftyMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
fiftvMSSEGac=median(num1);
/*print fiftvMSELR fiftvMSSEGac N;*/
create fiftyGac from fiftyMSSEGac[colname='Fifty' rowname='Gacula'];
   append from fiftyMSSEGac[rowname='Gacula'];
quit;


data FiftyGacA (rename= (_LIT1017=technique));
set FiftyGac;
run;


proc IML;
use twentyfifth;
read all var {_REF_} into real;
use Gac;
read all var {twentyfifth} into fifty;
/**print fifty real;*/


N=nrow(fiftv):
fiftyP=j(N,1,.);
```

```
do i=1 to N;
      fiftyP[j,1]=real;
end;
twentyfifthMSE=fiftvP||fiftv:
num=(twentyfifthMSE[,1]-twentyfifthMSE[,2]);

num2=num`*num;
den=N-1;
twentyfifthMSELRa=num2/den;
twentyfifthMSEGac=sum(twentyfifthMSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
twentyfifthMSSEGac=median(num1);
/*print twentyfifthMSEGac twentyfifthMSSEGac N;*/
create TwentyfifthGac from twentyfifthMSSEGac[colname='Twentyfifth' rowname='Gacula'];
   append from twentyfifthMSSEGac[rowname='Gacula'];
quit;


data TwentyfifthGacA (rename= (_LIT1017=technique));
set TwentyfifthGac;
run;



proc IML;
use tenth;
read all var {_REF_} into real;
use Gac;
read all var {tenth} into fifty;
/**print fifty real;*/


N=nrow(fiftv):
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
tenthMSE=fiftvP||fiftv:
num=(tenthMSE[,1]-tenthMSE[,2]);

num2=num`*num;
den=N-1;
tenthMSELRa=num2/den;
tenthMSEGac=sum(tenthMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
tenthMSSEGac=median(num1);
/*print tenthMSEGac tenthMSSEGac N;*/
create TenthGac from TenthMSSEGac[colname='Tenth' rowname='Gacula'];
   append from TenthMSSEGac[rowname='Gacula'];
quit;


data TenthGacA (rename= (_LIT1017=technique));
set TenthGac;
run;


proc IML;
use first;
read all var {_REF_} into real;
use Gac;
read all var {first} into fifty;
/**print fifty real;*/


N=nrow(fiftv):
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
firstMSE=fiftvP||fiftv:
num=(firstMSE[,1]-firstMSE[,2]);
```

```
num2=num`*num;
den=N-1;
firstMSELRa=num2/den;
firstMSEGac=sum(firstMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
firstMSSEGac=median(num1);
/*print firstMSEGac firstMSSEGac N;*/
create FirstGac from FirstMSSEGac[colname='First' rowname='Gacula'];
   append from FirstMSSEGac[rowname='Gacula'];
quit;


data FirstGacA (rename= (_LIT1017=technique));
set FirstGac;
run;



proc IML;
use fiftieth;
read all var {_REF_} into real;
use Gac;
read all var {fiftieth} into fifty;
/**print fifty real;*/

N=nrow(fifty);
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
fiftyMSE=fiftyP||fifty;
num=(fiftyMSE[,1]-fiftyMSE[,2]);

num2=num`*num;
den=N-1;
fiftyMSELRa=num2/den;
fiftyMSEGac=sum(fiftyMSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSEGac=median(num1);


quit;

proc IML;
use fiftieth;
read all var {_REF_} into real;
use Sloan;
read all var {fiftieth} into fifty;
/**print fifty real;*/

N=nrow(fifty);
fiftyP=j(N,1,.);

do i=1 to N;
      fiftyP[j,1]=real;
end;
MSE=fiftyP||fifty;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSESloan=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSESloan=median(num1);
/*print MSESloan MedSSESloan N;*/

create FiftySloan from MedSSESloan[colname='Fifty' rowname='Sloan'];
   append from MedSSESloan[ rowname='Sloan'];

quit;
```

```
data FiftySloanA (rename= (_LIT1017=technique));
set FiftySloan;
run;



proc IML;
use Twentyfifth;
read all var {_REF_} into real;
use Sloan;
read all var {twentyfifth} into Twentyfifth;
/**print fifty real;*/

N=nrow(Twentyfifth);
TwentyfifthP=j(N,1,.);

do j=1 to N;
      TwentyfifthP[j,1]=real;
end;
MSE=TwentyfifthP||Twentyfifth;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSESloan=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSESloan=median(num1);
/*print MSESloan MedSSESloan N;*/

create TwentyfifthSloan from MedSSESloan[colname='Twentyfifth' rowname='Sloan'];
   append from MedSSESloan[ rowname='Sloan'];

quit;

data TwentyfifthSloanA (rename= (_LIT1017=technique));
set TwentyfifthSloan;
run;


proc IML;
use Tenth;
read all var {_REF_} into real;
use Sloan;
read all var {Tenth} into Tenth;
/**print fifty real;*/

N=nrow(Tenth);
TenthP=j(N,1,.);

do j=1 to N;
      TenthP[j,1]=real;
end;
MSE=TenthP||Tenth;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSESloan=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSESloan=median(num1);
/*print MSESloan MedSSESloan N;*/

create TenthSloan from MedSSESloan[colname='Tenth' rowname='Sloan'];
   append from MedSSESloan[ rowname='Sloan'];

quit;

data TenthSloanA (rename= (_LIT1017=technique));
set TenthSloan;
run;


proc IML;
```

```
use First;
read all var {_REF_} into real;
use Sloan;
read all var {First} into First;
/**print fifty real;*/

N=nrow(First);
FirstP=j(N,1,.);

do j=1 to N;
      FirstP[j,1]=real;
end;
MSE=FirstP||First;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSESloan=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSESloan=median(num1);
/*print MSESloan MedSSESloan N;*/

create FirstSloan from MedSSESloan[colname='First' rowname='Sloan'];
   append from MedSSESloan[ rowname='Sloan'];

quit;

data FirstSloanA (rename= (_LIT1017=technique));
set FirstSloan;
run;


proc IML;
use fiftieth;
read all var {_REF_} into real;
use Lab;
read all var {fiftieth} into fifty;

N=nrow(fifty);
fiftyP=j(N,1,.);

do j=1 to N;
      fiftyP[j,1]=real;
end;
MSE=fiftyP||fifty;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSELab=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSELab=median(num1);
/*print MSELab MedSSELab N;*/

create FiftyLab from MedSSELab[colname='Fifty' rowname='Labuza'];
   append from MedSSELab[ rowname='Labuza'];

quit;

data FiftyLabA (rename= (_LIT1017=technique));
set FiftyLab;
run;


proc IML;
use Twentyfifth;
read all var {_REF_} into real;
use Lab;
read all var {twentyfifth} into Twentyfifth;

N=nrow(Twentyfifth);
TwentyfifthP=j(N,1,.);
```

```
do j=1 to N;
      TwentyfifthP[j,1]=real;
end;
MSE=TwentyfifthP||Twentyfifth;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSELab=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSELab=median(num1);
/*print MSELab MedSSELab N;*/

create TwentyfifthLab from MedSSELab[colname='Twentyfifth' rowname='Labuza'];
   append from MedSSELab[ rowname='Labuza'];

quit;

data TwentyfifthLabA (rename= (_LIT1017=technique));
set TwentyfifthLab;
run;



proc IML;
use Tenth;
read all var {_REF_} into real;
use Lab;
read all var {Tenth} into Tenth;

N=nrow(Tenth);
TenthP=j(N,1,.);

do j=1 to N;
      TenthP[j,1]=real;
end;
MSE=TenthP||Tenth;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
MSELRa=num2/den;
MSELab=sum(MSELRa);
num1=j(N,1,.);
do j=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSELab=median(num1);
/*print MSELab MedSSELab N;*/

create TenthLab from MedSSELab[colname='Tenth' rowname='Labuza'];
   append from MedSSELab[ rowname='Labuza'];

quit;

data TenthLabA (rename= (_LIT1017=technique));
set TenthLab;
run;


proc IML;
use First;
read all var {_REF_} into real;
use Lab;
read all var {First} into First;

N=nrow(First);
FirstP=j(N,1,.);

do j=1 to N;
      FirstP[j,1]=real;
end;
MSE=FirstP||First;
num=(MSE[,1]-MSE[,2]);

num2=num`*num;
den=N-1;
```

```
MSELRa=num2/den;
MSELab=sum(MSELRa);
num1=j(N,1,.);
do i=1 to N;
num1[j,1]=(num[j,1]*num[j,1]);
end;
MedSSELab=median(num1);
/*print MSELab MedSSELab N;*/

create FirstLab from MedSSELab[colname='First' rowname='Labuza'];
   append from MedSSELab[ rowname='Labuza'];

quit;

data FirstLabA (rename= (_LIT1017=technique));
set FirstLab;
run;


data MedMLE;
merge fiftyMLEA twentyfifthMLEA tenthMLEA firstMLEA ;
run;

data GacMLE;
merge FiftyGacA TwentyfifthGacA TenthGacA FirstGacA;
run;

data SloanMLE;
merge FiftySloanA TwentyfifthSloanA TenthSloanA FirstSloanA;
run;

data LabMLE;
merge FiftyLabA TwentyfifthLabA TenthLabA FirstLabA;
run;

data CMLE;
set  GacMLE LabMLE SloanMLE MedMLE;
run;



filename odsout 'C:\Documents and Settings\localuser\Desktop\Boxplots';
goptions device=gif;

/* close the listing destination */
ods listing close;
ods rtf path=odsout body='Fiftieth.rtf';
title1  "Fiftieth Percentile Estimates";
title2  &alpha1 &beta1 &nsamp1 &design;

proc boxplot data=combinedfifty ;
plot (fiftieth)*_model_/ cframe=vligb
                                                    cboxes= daqr
                                                    cboxfill = ywh
                                                     BOXSTYLE=SCHEMATIC


vref=fiftieth
VREFLABPOS=3;
;
insetgroup mean n;
run;


ods rtf close;
ods listing;
ods graphics off;

data combinedtenth;
set Gac Lab Sloan lrtenth;
/**by Iteration;*/
run;


data combinedtwentyfifth;
set Gac Lab Sloan lrtwentyfifth;
/**by Iteration;*/
run;


filename odsout 'C:\Documents and Settings\localuser\Desktop\Boxplots';
goptions device=gif;
```

```
/* close the listing destination */
ods listing close;
ods rtf path=odsout body='Twentyfifth.rtf';
title1  "Twentyfifth Percentile Estimates";
title2
color=black &alpha1 &beta1 &nsamp1 &design;

proc boxplot data=combinedtwentyfifth;
plot (twentyfifth)*_model_/ cframe=vligb
                                                    cboxes= dagr
                                                    cboxfill = ywh
                                                            BOXSTYLE=SCHEMATIC
vref=twentyfifth
VREFLABPOS=3;
insetgroup mean n;
run;

ods rtf close;
ods listing;
ods graphics off;
data _null_;
run;

filename odsout 'C:\Documents and Settings\localuser\Desktop\Boxplots';
goptions device=gif;

ods rtf path=odsout body='Tenth.rtf';

title1  "Tenth Percentile Estimates";
title2
color=black &alpha1 &beta1 &nsamp1 &design;



proc boxplot data=combinedtenth;
plot (tenth)*_model_/ cframe=vligb
                                                    cboxes= dagr
                                                    cboxfill = ywh
                                                            BOXSTYLE=SCHEMATIC
vref=tenth
VREFLABPOS=3;
insetgroup mean n;
run;


ods rtf close;
ods listing;
ods graphics off;



data combinedfirst;
set Gac Lab Sloan lrfirst;
/**by Iteration;*/
run;

ods rtf close;
ods listing;
ods graphics off;

filename odsout 'C:\Documents and Settings\localuser\Desktop\Boxplots';
goptions device=gif;

/* close the listing destination */
ods listing close;

ods rtf path=odsout body='1st.rtf';

title1  "First Percentile Estimates";
title2
color=black &alpha1 &beta1 &nsamp1 &design;

/*title 'First Percentile Estimates';*/
proc boxplot data=combinedfirst;
/**where first < 1000;*/

plot (first)*_model_/ cframe=vligb
                                                    cboxes= dagr
                                                    cboxfill = ywh
                                                    BOXSTYLE=SCHEMATIC
vref=first
```

```
/*alllabel=VALUE*/
/*labelangle=90*/
/*clabel=red*/

/*VREFLABELS='Fifty';*/
VREFLABPOS=3;
insetgroup mean n;
/*vzero;*/
run;

ods rtf close;
ods listing;
ods graphics off;
/**/
/*Nice graph section*/


filename odsout 'C:\Documents and Settings\localuser\Desktop\Final Graphs';
goptions device=gif;

/* close the listing destination */
ods listing close;


ods rtf path=odsout body='MedSSEtest.rtf';
title Median SSE &alpha1 &beta1 &nsamp1 &design;


data CMLE;
set CMLE:
Fifty=round(Fifty,.01);
Twentyfifth=round(Twentyfifth,.01);
Tenth=round(Tenth,.01);
First=round(First,.01);
run;

proc print data=CMLE noobs;
run;
ods rtf close;
ods listing;
ods graphics off;

data SloanCoverage (rename=(  Sloancoveragefifty=Fiftieth   SloancoveragetwentyFive=TwentyFifth
SloancoverageTenth=Tenth   SloancoverageFirst=First));
merge nameSloan Sloan50Dc Sloan25Dc Sloan10Dc Sloan01Dc;
run;


data LabCoverage (rename=( Labcoveragefifty=Fiftieth  LabcoveragetwentyFive=TwentyFifth
LabcoverageTenth=Tenth  LabcoverageFirst=First));
merge nameLabuza Lab50Dc Lab25Dc Lab10Dc Lab01Dc;
run;

data coverageLF (rename=( MLEcoveragefifty=Fiftieth  MLEcoveragetwentyFive=TwentyFifth
MLEcoverageTenth=Tenth  MLEcoverageFirst=First)); ;
merge nameMLE c50F c25F c10F  c1F  ;
run;

data GacCoverage (rename=(Gaccoveragefifty=Fiftieth GaccoveragetwentyFive=TwentyFifth
GaccoverageTenth=Tenth GaccoverageFirst=First));
merge nameG Gc50Dc Gc25Dc Gc10Dc Gc01Dc;
run;

filename odsout 'C:\Documents and Settings\localuser\Desktop\Final Graphs';
goptions device=gif;

/* close the listing destination */
ods listing close;


ods rtf path=odsout body='Coverage.rtf';
title Percent Coverage &alpha1 &beta1 &nsamp1 &design;

data OCoverage;
set GacCoverage LabCoverage SloanCoverage coverageLF;
Fiftieth=round(Fiftieth,.01);
Twentyfifth=round(Twentyfifth,.01);
Tenth=round(Tenth,.01);
First=round(First,.01);
run;
proc print data=OCoverage noobs;
```

```
run;

ods rtf close;
ods listing;
ods graphics off;
%mend Sim;
/*%Sim(iter=1000,alpha=63.14,beta=7.17);*/
%let k=1;
%let alpha1=alpha(&alpha);
%let beta1=beta(&beta);
%let nsamp1=sample size(&nsamp);
%let design=Independent Run #2;
/*1*/
/*%Sim (iter=10000 ,alpha=65.80 ,beta=4.05 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*2*/
%Sim (iter=100 ,alpha=65.80 ,beta=4.05 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);
/*3*/
/*%Sim (iter=10000 ,alpha=65.80 ,beta=4.05 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*4*/
/*%Sim (iter=10000 ,alpha=65.80 ,beta=4.05 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*5*/
/*%Sim (iter=10000 ,alpha=63.14 ,beta=7.17 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*6*/
/*%Sim (iter=10000 ,alpha=63.14 ,beta=7.17 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*7*/
/*%Sim (iter=10000 ,alpha=63.14 ,beta=7.17 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*8*/
/*%Sim (iter=10000 ,alpha=63.14 ,beta=7.17 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*9*/
/*%Sim (iter=10000 ,alpha=66.92 ,beta=2.0 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*10*/
/*%Sim (iter=10000 ,alpha=66.92 ,beta=2.0 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*11*/
/*%Sim (iter=10000 ,alpha=66.92 ,beta=2.0 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*12*/
/*%Sim (iter=10000 ,alpha=66.92 ,beta=2.0 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*13*/
/*%Sim (iter=10000 ,alpha=67.00 ,beta=1.62 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*14*/
/*%Sim (iter=10000 ,alpha=67.00 ,beta=1.62 ,nsamp=50 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*15*/
/*%Sim (iter=10000 ,alpha=67.00 ,beta=1.62 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=14 ,int2=7 ,int3=3
,stsize=3,c=1,accelerate=.5);*/
/*16*/
/*%Sim (iter=10000 ,alpha=67.00 ,beta=1.62 ,nsamp=75 ,ntest=1 ,sttime=7 ,int1=7 ,int2=7 ,int3=7
,stsize=7,c=0,accelerate=2);*/
/*%inc 'c:/';*/
/*libname design '.';*/
/*%macro bsetup*/
/*let dsname=design;*/
```

Appendix 3: SAS code for the Genetic Algorithm

```
/*libname '.':*/
options ls=72;
options FORMDLIM="#";
options nosource nosource2 nonotes errors=100;
/*This step is for the PC, to write the log to a file, otherwise the log would
fill up with warnings*/

/*proc printto log='log.log';*/
/*run;*/

/*This is the Simulation that is used to calculate the mese's of the sampling designs*/

%macro Sim (iter= ,alpha= ,beta= ,nsamp= ,ntest= ,sttime= ,int1= ,int2= ,int3= ,stsize= ,c=
,accelerate= ,percentile= );

proc iml;
/**  define parameters;*/

/**  nit is the number of iterations;*/
nit= &iter;


/**setting up the real Weibull distribution;*/
alpha= &alpha ;
/*63.14;*/
beta= &beta;
/**7.17;*/
/*alpha=50;
beta=1.2;
*/
/**find the median of the Weibull;*/
whatever=alpha*((-log(&percentile))**(1/beta));
med=alpha*((-log(.5))**(1/beta));
first=alpha*((-log(.99))**(1/beta));
tenth=alpha*((-log(.9))**(1/beta));
twentyfifth=alpha*((-log(.75))**(1/beta));
fiftieth=alpha*((-log(.5))**(1/beta));


/** for use in detecting coverage;*/
firstcol=j(nit,1,first);
tenthcol=j(nit,1,tenth);
twentyfifthcol=j(nit,1,twentyfifth);
fiftiethcol=j(nit,1,fiftieth);


/**setting up the total # we will ever sample;*/
nsamp=&nsamp;
/**25;*/

/**how many times we can test before censored;*/
ntest=&ntest;

/**when the first sample is taken;*/
sttime=&sttime;
/**14;*/
time=sttime;

/**for a loop later on about when to begin accelarated testing;*/
half=0;

/**How many days between samples before first failure;*/
int1= &int1;
/**14;*/

/**# of the days between samples after first failure;*/
int2= &int2;
/*7*/

/**# of days between samples after half the samples have failed;*/
int3= &int3;
/*3*/

/**random #;*/
seed=1;

/**n for first sample at sttime;*/
stsize= &stsize;

dumsize=stsize;
   time=sttime;
```

```
sint1=&c;
sint2=&c;
sint3=&c;
a=&accelerate;

/**  start iterations;*/
do i=1 to nit;

/****data=j(nsamp,5,-9);*/
   data=j(nsamp,7,-9);
/*   * so that we can put iteration number in the array for later use in a by statement;*/
   cfail=0;
   tfail=0;

dumsize=stsize;
time=sttime;
/**for coverage later;*/
cover1=j(4,1,.);
c= {.002180575 .002180575  .002180575,
 1.02268  1.02268 1.02268,
 1.45891  1.45891 1.45891,
 1.84083  1.84083 1.84083 };
 do j=1 to 4;
 cover1[j,1]=i;
 end;
 cover2=c||cover1;


;
/**  generate data:*/
   do j=1 to nsamp;
      data[j,1]=rand('weibull', beta, alpha);  %*** the actual failure time of the unit;
           data[j,2]=0;                             %*** number of times the unit has been tested;
           data[j,3]=0;                             %*** contains the censoring variable (0,1,2);

                                                                                                  %***
column 4 the observed failure or censoring time of the unit;

*     data[j,5]=ranuni(0);                    %*** random numbers for random sorting and sampling of
the nonfailed units units;
    data[j,6]=.;                             %*** for storage of next to last observed value;
        data[j,7]=i;                                                  %*** iteration number;
        end;

   done=0;

   do until (done>1);


      do j=1 to nsamp;
           data[j,5]=ranuni(seed);
         end;

/**assigns code so that censored and failed units will move to the end of the array;*/
      do j=1 to nsamp;
         if data[j,3]>0
         then data[j,5]=2;
      end;


/**sorts data by the random #'s for sampling;*/
      scdata=rank(data[,5]);
      dum=data;
      do w= 1 to 6;
         data[scdata,w]=dum[,w];
      end;

/**** print currentsize time data;*/

sfail=0;

/**stops resampling failed or censored units and samples;*/
      currentsize=dumsize;
if currentsize>nsamp
then currentsize=nsamp;
      do j=1 to currentsize;

/****I think sfail is the number of failures in this sample, cfail is the cumulative number of failues
in this study,*/
/*               and tfail is an indicator of whether there were multiple failures in this sample;*/
```

```
         if data[j,3]>0
           then goto here;
           else
              data[j,2]=data[j,2]+1;


         if data[j,1]<time
               then
               do:
                  data[j,4]=time;
                  data[j,3]=1;
                  cfail=cfail+1;
                  sfail=sfail+1;
              if sfail>1
                then tfail=1;
               end;
               else
                       if data[j,2]=ntest
                    then
                          do:
                            data[j,4]=time;
                            data[j,3]=2;

                                       end;
/*               *this sets up the beginning part of the interval censored data when the sampling is
not destructive;*/
                           if data[j,2]<ntest & data[j,4]<data[j,1]
                           then
                             do:
                                data[j,6]=time;
                                end;




         here:    ;


      end;


if cfail=0 then goto there;


if sfail >= (currentsize*a) & half=0
         then do;

                 half=1;

                 end;

if  half=0
         then do;
         time=time+int2;
                 dumsize=currentsize+sint2;
                 end;
if half=0 then goto hither;

if half=1
         then do;
                 time=time+int3;
                 half=1;
                 dumsize=currentsize+sfail+sint3;
                 end;
hither: :
if cfail ^=0 then goto thither;
 there:        do;
         time=time+int1;
                  dumsize=currentsize+sint1;
              end;
thither: ;

/**current group:*/
   done=min(data[,5]);



   end;
```

```
rankdata=1+nsamp-rank(data[,4]);
hazdata=j(nsamp,1);
sumhaz=0;
cumhaz=j(nsamp,1);
do ijk=1 to nsamp;

if data[ijk,3]=1
then
hazdata[ijk]=1/rankdata[ijk];
else hazdata[ijk]=.;


end:
masterdat=rankdata||hazdata||(data[,4])||(data[,1]);
     call sort( masterdat, {1 4}, {1} );
do ijk=1 to nsamp;
 if masterdat[ijk,2] ^= .
 then do;
cumhaz[ijk]=sumhaz+masterdat[ijk,2];
sumhaz=cumhaz[ijk]; end;
 else cumhaz[ijk]=.;
end;



begin=j(nsamp,1,.);
end=j(nsamp,1,.);
del=j(nsamp,1,.);
sumdel=0;
cumdel=j(nsamp,1,.);
do ijk=1 to nsamp;
 if data[ijk,3] = 1
  then begin[ijk]= data[ijk,6];
   else begin[ijk]=data[ijk,4];
 if data[ijk,3] = 1
  then end[ijk]=data[ijk,4];
   else end[ijk]=.;
end;

do ijk=1 to nsamp;
 if end[ijk]^=.
   then del[ijk]=1;
end;

do ijk=1 to nsamp;
if end[ijk]^=.
then do;
cumdel[ijk]=sumdel+del[ijk];
 sumdel=cumdel[ijk]; end;
 else cumdel[ijk]=.;
end:
maxdel=j(nsamp,1,.);

do ijk=1 to nsamp;
  maxdel[ijk]=max(cumdel);
end:
  cmaxdel=j(4,1,.);

do ijk=1 to 4;
 cmaxdel[ijk]=max(cumdel);
end;



iter=j(nsamp,1,i);
Control=j(nsamp,1,.);









Perc=first||tenth||twentvfifth||fiftieth||whatever;
Life=begin||end||iter||data[,1]||maxdel;


if nit=1 then Life1=Life;
else Life1=Life1//Life;
```

```
end;


create Percentile from Perc;
append from Perc;

create Reg from Life1;
append from Life1;

quit;



data Lifereg (rename= (COL1=begin COL2=end COL3=iter COL4=real COL5=del));
set Reg;
control=0;
retain dum 0;
if COL3 ne dum then control=1;
dum=COL3;
if COL2=1000 then COL2=.;
run;


proc lifereg data=Lifereg noprint;
by iter;
model (begin, end)= /  maxiter=50  ;
output out=d2 quantiles=.01 .1 .25 .5 std=std p=predtime control=control  ;
run;


data lrfifty;
set d2;
 Model ='Lifereg';
if _PROB_ ^= .50
then delete;
fiftieth=predtime;
run;


data Percentile (rename= (COL1=first COL2=tenth COL3=twentyfifth COL4=fiftieth COL5=Percent));
set Percentile;
run;

data Percent;
set Percentile;
 REF_=Percent;
_REFLAB_='Variable';
keep _REF_ _REFLAB_;
run;


data lrfiftyclean;
set lrfiftv;
if fiftieth=. then delete;
run;

proc IML;
use Percent;
read all var { REF_} into real;
use lrfiftyclean;
read all var {fiftieth} into fifty;

N=nrow(fiftv);
fiftyP=j(N,1,.);

do j=1 to N;
      fiftyP[j,1]=real;
end;
MSE=fiftvP||fiftv;
num=(MSE[,1]-MSE[,2]);
num2=.;
num2=num`*num;
if num2=. then num2=15000;
den=N-1;
MSELRa=num2/den;

num1=j(N,1,.);
do i=1 to N;
num1[i,1]=(num[i,1]*num[i,1]);
if num1[j,1]=. then num1[j,1]=1000;
end;
```

```
MSEmed=median(num1);


MSE=sum(MSELRa);
Optimal=MSE||MSEmed;
create M from Optimal;
append from Optimal;

quit;

data Optimal (rename= (COL1=MSE COL2=MedSSE));
set M;
run;

data results; set results Optimal;
%mend Sim;


/*These three macro 'loops' feeds the sampling design values into the simulation
Loop is for generation 0
LoopM is for the mutation
LoopC is for the crossover*/


%macro loop(nr);
%*This macro reads a line from the sample file, converts the data step variables v1-vn
a into macro variables and calls the macro sim. It repeats this operation nr times;
%let n=%eval(&nr);
%let i=1;
%do %until(%eval(&i) > %eval(&n));
%let numr=%eval(&i);
%put 'numr' &numr;
data f&numr ; set cover;
if _n_=&numr;
run;

%*The next data step puts the data step variables from the numr line of the file
 sample and puts them into the macro variables a-pe;
data  null ; set f&numr;
call symput('a',trim(left(initialsample)));
call symput('b',trim(left(timefirstsample)));
call symput('c',trim(left(C)));
call symput('d',trim(left(delta1)));
call symput('e',trim(left(delta2)));
call symput('f',trim(left(delta3)));
call symput('h',trim(left(accelerate)));
call symput('j',trim(left(nsamp)));
call symput('la',trim(left(alpha)));
call symput('ma',trim(left(beta)));
call symput('pe',trim(left(perc)));
run;

%*now it calls macro Sim with the various arguments:
%Sim (iter=1000 ,alpha=&la ,beta=&ma ,nsamp=&j ,ntest=1 ,sttime=&a ,int1=&b ,int2=&c ,int3=&d
,stsize=&e ,c=&f,accelerate=&h,Percentile=&pe);
%let i=%eval(&i+1);
%end;
%mend loop;


%macro loopM(nr);
%*This macro reads a line from the sample file, converts the data step variables v1-vn
a into macro variables and calls the macro sim. It repeats this operation nr times;
%let n=%eval(&nr);
%let i=1;
%do %until(%eval(&i) > %eval(&n));
%let numr=%eval(&i);
%put 'numr' &numr;
data f&numr ; set mutation1;
if _n_=&numr;

%*The next data step puts the data step variables from the numr line of the file
 sample and puts them into the macro variables a-pe;
data  null ; set f&numr;
call symput('a',trim(left(initialsample)));
call symput('b',trim(left(timefirstsample)));
call symput('c',trim(left(C)));
call symput('d',trim(left(delta1)));
call symput('e',trim(left(delta2)));
call symput('f',trim(left(delta3)));
call symput('h',trim(left(accelerate)));
```

```
call symput('j',trim(left(nsamp)));
call symput('la',trim(left(alpha)));
call symput('ma',trim(left(beta)));
call symput('pe',trim(left(perc)));
run;

%*now it calls macro Sim with the various arguments;
%Sim (iter=1000 ,alpha=&la ,beta=&ma ,nsamp=&j ,ntest=1 ,sttime=&a ,int1=&b ,int2=&c ,int3=&d
,stsize=&e ,c=&f,accelerate=&h,Percentile=&pe);
%let i=%eval(&i+1);
%end:
%mend loopM;


%macro loopC(nr);
%*This macro reads a line from the sample file, converts the data step variables v1
and v2 into macro variables and calls the macro ap. It repeats this operation nr times;
%let n=%eval(&nr);
%let i=1;
%do %until(%eval(&i) > %eval(&n));
%let numr=%eval(&i);
%put 'numr' &numr;
data f&numr ; set crossover1;
if _n_=&numr;
run;
%*The next data step puts the data step variables from the numr line of the file
 sample and puts them into the macro variables a-pe;
data  null ; set f&numr;
call symput('a',trim(left(initialsample)));
call symput('b',trim(left(timefirstsample)));
call symput('c',trim(left(C)));
call symput('d',trim(left(delta1)));
call symput('e',trim(left(delta2)));
call symput('f',trim(left(delta3)));
call symput('h',trim(left(accelerate)));
call symput('j',trim(left(nsamp)));
call symput('la',trim(left(alpha)));
call symput('ma',trim(left(beta)));
call symput('pe',trim(left(perc)));
run;
%*now it calls macro Sim with the various arguments;
%Sim (iter=1000 ,alpha=&la ,beta=&ma ,nsamp=&j ,ntest=1 ,sttime=&a ,int1=&b ,int2=&c ,int3=&d
,stsize=&e ,c=&f,accelerate=&h,Percentile=&pe);
%let i=%eval(&i+1);
%end:
%mend loopC;


/*This macro loops the Mutation and Crossover subroutines G times,*/
/*the number of generations*/
%macro overallA(G, M= ,nsamp= ,alpha1= ,beta1= ,perc1= ,);
%let g=%eval(&G);
%let k=0;

%do %while (&k<&G+1);

%put the value of k is &k;



%if %eval(&k)>0 %then %goto here;

data combine&G;
run;

/*This step generates the generation 0 data within the bounds set*/
proc iml;
M=%eval(&M);
nsamp=%eval(&nsamp);
alpha=%eval(&alpha1);
/*66.14;*/
beta=%eval(&beta1);
/*4.05:*/
Perc=1/%eval(&perc1);
initial= j(M,11,1);
pre=j(M,3,1);
do j=1 to M;
initial[j,1]=round(rand('uniform')*.1*nsamp,1);
if initial[j,1]=0 then initial[j,1]=1;
initial[j,2]=round(alpha*((-log(1-(rand('uniform'))*.1*Perc))**(1/beta)),1);
initial[j,3]=round(rand('uniform')*5,1);
```

```
initial[j,7]=(rand('uniform')*Perc)+.1*Perc;
initial[j,8]=nsamp;
initial[j,9]=alpha;
initial[j,10]=beta;
initial[j,11]=Perc;
end:
do j=1 to M;
pre[j,1]=(rand('uniform')*9+1)/100;
pre[j,2]=(rand('uniform')*9+1)/100;
pre[j,3]=(rand('uniform')*9+1)/100;
end:
do j=1 to M;
initial[j,4]=round(alpha*((-log(1-(rand('uniform')*.49*Perc+.01)))**(1/beta)),1);
initial[j,5]=round(alpha*((-log(1-(rand('uniform')*.49*Perc+.01)))**(1/beta)),1)-round(alpha*((-log(1-
.01))**(1/beta)),1);
if initial[j,5]<1 then initial[j,5]=1;
initial[j,6]=round(alpha*((-log(1-(rand('uniform')*.49*Perc+.01)))**(1/beta)),1)-round(alpha*((-log(1-
.1))**(1/beta)),1);
if initial[j,6]<1 then initial[j,6]=1;
end;

/*print pre initial;*/
create cover from initial;
append from initial;
quit;

data cover (rename= (COL1=initialsample COL2=timefirstsample COL3=C COL4=delta1 COL5=delta2
COL6=delta3 COL7=accelerate COL8=nsamp COL9=alpha COL10=beta COL11=Perc));
set cover;
run;


data results;
:
%loop(&M)


data results; set results;
if _n_>1;
data combine; merge cover results;
proc iml;
use combine;
read all var {MedSSE} into X;
n=nrow(X);
S=Sum(X);
first=S/n;
create genzero from first;
append from first;
quit;
data genzero (rename=(COL1=genzero));
set genzero;
run;


%here : ;
proc sort data=combine;
by MedSSE;
run;
data combineA;
set combine;
Size=0;
do start=1 to _n_;
Size=Size+1;
retain Size;
end:
Size1=1/Size;

run;



/**/
/*Mutation Subroutine*/
/**/

/*Here is the code for the mutatation part of the genetic algorithm;*/

proc IML;
use combineA;
nsamp=%eval(&nsamp);
alpha=%eval(&alpha1);
```

```
beta=%eval(&beta1);
a=2;
Perc=1/%eval(&perc1);
read all var {initialsample timefirstsample C delta1 delta2 delta3 accelerate nsamp} into X;
n=nrow(X);
G=%eval(&k);
T=.1;
M=%eval(&M);
parm=j(M,3,-9);
do i=1 to M;
parm[i,1]=alpha;
parm[i,2]=beta;
parm[i,3]=Perc;
end;
Rand=j(M,7,-9);
do i=1 to M;
 do j=1 to 7;
 Rand[i,j]=rand('Uniform');
 end;
end;
Prob=j(M,7,-9);
do i=1 to M;
 do j=1 to 7;
 Prob[i,j]=1/exp(G*T);
 end;
end;



/*Setting up maxes for Mutation*/

nmax=round(nsamp*.1,1);
maxitime=round(alpha*((-log(1-.1*Perc))**(1/beta)),1);
maxdelta1=round(alpha*((-log(1-.5*Perc))**(1/beta)),1);
maxdelta2=round(alpha*((-log(1-.5*Perc))**(1/beta)),1);
maxdelta3=round(alpha*((-log(1-.5*Perc))**(1/beta)),1);
maxaccelerate=.8*Perc;

XM=j(M,7,-9);
XM=X;
do i=1 to M;
 do j=1 to 7;
  if Rand[i,j]<Prob[i,j] then goto mutate;
   else goto here;
   mutate:

if j=1 then do;
   p=X[i,j]/nmax;
   if p=1 then p=.90;
   if p>1 then p=.90;
   if p<0 then p=.1;
   if p=0 then p=.1;
   if nmax=0 then nmax=2;
   if nmax<=1 then nmax=2;
   XM[i,j]=rand('Binomial',p,nmax-1)+1;
   end;

if j=2 then do;
   p=X[i,j]/maxitime;
   if maxitime=0 then maxitime=2;
   if maxitime=1 then maxitime=2;
    if p=1 then p=.90;
   if p>1 then p=.90;
   if p<0 then p=.1;
   if p=0 then p=.1;
   XM[i,j]=rand('Binomial',p,maxitime-1)+1;
   end;

if j=3 then do;
   p=X[i,j]/5;
 if p=1 then p=.90;
   if p>1 then p=.90;
   if p<0 then p=.1;
   if p=0 then p=.1;
   XM[i,j]=rand('Binomial',p,5);
   end;

if j=4 then do;
   p=X[i,j]/maxdelta1;
   if p=1 then p=.90;
   if p>1 then p=.90;
```

```
   if p<0 then p=.1;
   if p=0 then p=.1;

   if maxdelta1<=1 then maxdelta1=2;
   XM[i,j]=rand('Binomial',p,maxdelta1-1)+1;
   end;


if j=5 then do;
   p=X[i,j]/maxdelta2;
    if p=1 then p=.90;
   if p>1 then p=.90;
   if p<0 then p=.1;
   if p=0 then p=.1;
   if maxdelta2<=1 then maxdelta2=2;
   XM[i,j]=rand('Binomial',p,maxdelta2-1)+1;
   end;

if j=6 then do;
   p=X[i,j]/maxdelta2;
    if p=1 then p=.90;
   if p>1 then p=.90;
   if p<0 then p=.1;
   if p=0 then p=.1;

   if maxdelta3<=1 then maxdelta3=2;
   XM[i,j]=rand('Binomial',p,maxdelta3-1)+1;
   end;

if j=7 then do;
   b=(2-2*X[i,j])/X[i,j];
    if b=0 then b=1;
   if b<0 then b=1;
   XM[i,j]=rand('Beta',a,b);
   end;
   here:
  end;
end;
XM=XM||parm;

create Mutation from XM;
append from XM;
quit;
data Mutation1 (rename= (COL1=initialsample COL2=timefirstsample COL3=C COL4=delta1 COL5=delta2
COL6=delta3 COL7=accelerate COL8=nsamp COL9=alpha COL10=beta COL11=Perc));
set Mutation;
run;


data results;
run:
%loopM(&M)
data results; set results;
if _n_>1;
data mutationresults; merge mutation1 results;

proc sort data=mutationresults;
by MedSSE;
run;
data mutationresultsA;
set mutationresults;
Size=0;
do start=1 to _n_;
Size=Size+1;
retain Size;
end:
Size1=1/Size;

run;



/**/
/*Crossover subroutine*/
/**/
proc surveyselect data=CombineA method=PPS  Sampsize=2 out=C2 noprint;
Size Size1;
run;

proc IML;
use C2;
```

```
alpha=%eval(&alpha1);
beta=%eval(&beta1);
nsamp=%eval(&nsamp);
read all var {initialsample timefirstsample C delta1 delta2 delta3 accelerate Perc} into X;
X1=X`;
n=nrow(X1);


N1=%eval(&M)*n:
new=j(N1,1,-9);
rand=j(N1,1,-9);
do i=1 to N1;
rand[i,1]=rand('uniform');
end:
do i=1 to %eval(&M);
if i=1 then X2=X1;
else X2=X2//X1;
end;
X2=X2||rand||new;
do i=1 to N1;
if X2[i,3]>.5 then X2[i,4]=X2[i,1];
else X2[i,4]=X2[i,2];
end:
do i=1 to %eval(&M);
j=i*7;
k=j-6;
X5=X2(|k:j,4|);
if i=1 then X6=X5`;
else X6=X6//X5`;
end:
X7=j(%eval(&M),4,-9);
do i=1 to %eval(&M);
X7[i,1]=nsamp;
X7[i,2]=alpha;
X7[i,3]=beta;
X7[i,4]=X[1,8];
end;
X6=X6||X7;
create crossover from X6;
append from X6;
quit;

data crossover1 (rename= (COL1=initialsample COL2=timefirstsample COL3=C COL4=delta1 COL5=delta2
COL6=delta3 COL7=accelerate COL8=nsamp COL9=alpha COL10=beta COL11=perc));
set crossover;
run;




data results;
run:
%loopC(&M)
data results; set results;
if _n_>1;
data combinecross; merge crossover1 results;
proc sort data=combinecross;
by MedSSE;
run;
data combinecrossA;
set combinecross;
Size=0;
do start=1 to _n_;
Size=Size+1;
retain Size;
end:
Size1=1/Size;
run;


data overall;
set combinecrossA mutationresultsA combineA;
Generation=0;
do Start1=1 to _n_;
Generation=%eval(&k);
end;
drop Size1 Start Size Start1;
run;

proc sort data=overall;
```

```
by MedSSE;
run;
data combine;
set overall;
M=%eval(&M);
Number=0;
do start=1 to _n_;
Number=Number+1;
retain Number;
end;
drop start;
if Number>M then delete;
run;

data combine&G;
set combine&G combine;
mese=MedSSE;
run;

%if %eval(&k)^=%eval(&G) %then %goto here1;

data ratio1;
set combine;
if _n_^=1 then delete;
keep MedSSE;
run;
data ratio2;
merge genzero ratio1;
R=1-(MedSSE/genzero);
keep R;
run;
data last;
set combine;
if _n_^=1 then delete;
run;
data lastone;
merge last ratio2;
run;

/*This last step generates the various output, which includes boxplots */
/*of the progress of the mese through the generations, and final mese*/
/*ratio of improvement, and parameter values of the design with the */
/*smallest mese at the last generation.*/

filename odsout '/home/vespers';
goptions device=gif;

ods listing close;
ods rtf path=odsout body='boxthousand2.rtf';

proc boxplot data=combine&G;
plot Mese*generation/ cframe=vligb
                                                    cboxes= dagr
                                                    cboxfill = ywh
                                                            BOXSTYLE=SCHEMATIC;
*insetgroup mean n;
run;

ods rtf close;
ods listing;
ods graphics off;

filename odsout '/home/vespers';
/*'/home/vespers';*/
goptions device=gif;

/* close the listing destination */
ods listing close;
ods rtf path=odsout body='thousand2.rtf';


proc print data=lastone;
run;

ods rtf close;
ods listing;
ods graphics off;

%here1 :;
%let k=%eval(&k+1);
%end;
```

```
%mend overallA;
/**/
/*This is the inputs for the macro, the first number is the number of generations*/
/*wanted, and so forth.*/

%overallA(5, M=50, nsamp=50, alpha1=63, beta1=7, perc1=2);
```

Bibliography

Cardelli, C., and Labuza, T. P. (2001), "Application of Weibull Hazard Analysis to the Determination of the Shelf Life of Roasted and Ground Coffee," *International Journal for Chemistry, Biochemistry, Microbiology and Technology of Food Processing*, 34 (5): 273-278.

Freitas, M. A., Wagner, B., Ho, L. L. (2003), "A Statistical Model for Shelf Life Estimation Using Sensory Evaluations Scores," *Communications in Statistics*, 32 (8): 1559-1589.

Gambaro, A., Fiszman, S., Gimenex, A., Varela, P., and Salvador, A. (2004), "Consumer Acceptability Compared with Sensory and Instrumental Measures of White Pan Bread: Sensory Shelf-life Estimation by Survival Analysis," *Journal of Food Science*, 69(9):401-405

Gacula, M. C. (1975), "The Design of Experiments for Shelf Life Study," *Journal of Food Science*, 40:399-403.

Gacula, M. C., and Singh, J. (1984). *Statistical models for shelf life failures.* New York: Academic Press Inc.

Gajewski, J. B., Sedwick, D. J., and Antonelli J. P. (2004) "A log-normal distribution model of the effect of bacteria and ear fenestration on hearing loss: a Bayesian approach" *Statistic in Medicine* 23:493-508

Hamada, M., Martz, H. F., Reese, C. S., and Wilson, A. G. (2001), "Finding Near-Optimal Bayesian Experimental Designs via Genetic Algorithms," *The American Statistician* 55(3):175-181.

Klein, P. J., Moeschberger, M., L. (1997) *Survival Analysis: Techniques for Censored and Truncated Data (Statistics for Biology and Health)*. Springer-Verlag Telos.

Labuza, T. P., Schmidl M. K. (1988), "Use of Sensory Data in the Shelf Life Testing of Foods: Principles and Graphical Methods for Evaluation," *Cereal Foods World* 33 (2):193-205.

Labuza, T. P., Schmidl, M. K. (1985), "Accelerated Shelf-Life Testing of Foods," *Food Technology* 39 (9):57-134.

Pawitan, Y. (2001) *Statistical Modelling and Inference Using Likelihood*. Oxford: Clarendon Press

Rencher, C. A. (2000). *Linear Models in Statistics*. New York: Wiley.

SAS Institute Inc. (1997), SAS/STAT® Macro Language: Reference, First edition, Cary, NC:SAS Institute Inc. 304pp.

SAS Institute Inc. (1999), SAS/STAT®. User's Guide, Version 8, Cary, NC: SAS Institute Inc. 3884pp.

Sloan, R. A. (2005), "Effect of water activity and packaging material on the quality of dehydrated taro slices during accelerated storage," M.S. Thesis, Department of Food Science and Nutrition, Brigham Young University

Tobias, P.A., Trinidade, D.C. (1995). *Applied Reliability* New York: Van Nostrand Reinhold 17(1):113-122

Wild, R., Collani, E. V., (2002). "Modeling of Explosives Sensitivity Part I: The Bruceton Method," *Economic Quality Control*